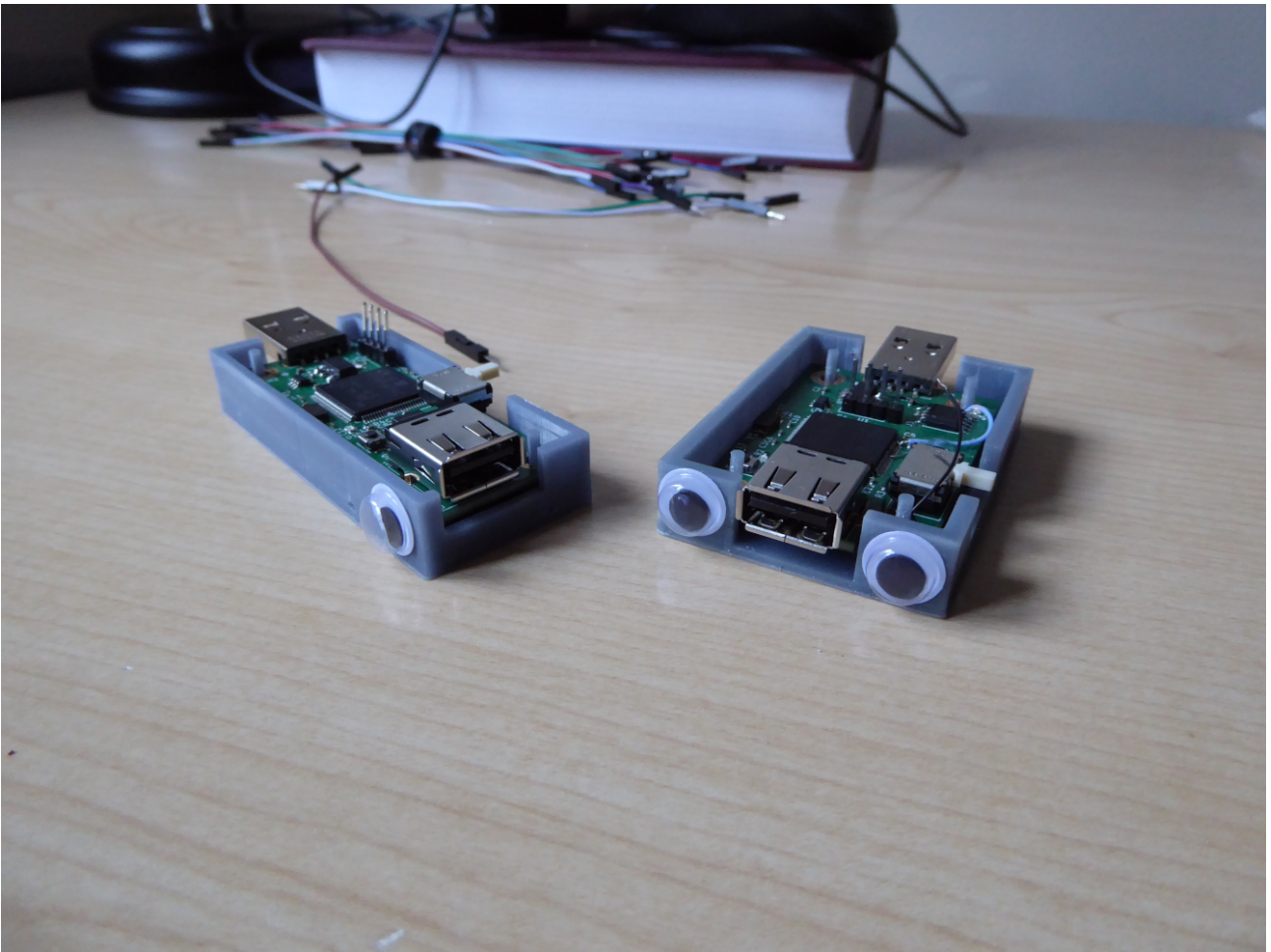


CKAKL

James Gavin, Bradley Gillies & Samuel Hunt
ACDA5017

July 11, 2025



Contents

1	Introduction	8
2	Literature Review	9
3	Research	11
3.1	Micro-controller	11
3.2	Flash Memory	11
3.3	Configuration File	12
3.4	Keyboard Features	12
3.5	Scan Codes	13
4	Requirements	14
5	Project Management	15
5.1	Methodology	15
5.2	Ethics	15
5.3	Gantt Chart	16
6	Hardware Design	18
6.1	Micro-controller	18
6.1.1	Review	20
6.2	Oscillator	20
6.3	USB Connection	21
6.3.1	Device	21
6.3.2	Host	21
6.3.3	Review	21
6.4	Voltage regulator	21
6.5	Programming Interface	21
6.6	Mode Switch	23
6.7	Flash Memory	23
6.8	Reset Button	24
6.9	LED indicator	24
6.10	Finished Schematic	25
6.11	PCB Design	26
6.12	Production	28
6.13	Case Design	28
6.14	Pinout	28
7	Hardware Testing	30
7.1	Power On	30
7.2	SWD	30
7.3	Clock	31
7.4	LED	31
7.5	Switch	31
7.6	Serial	33
7.7	Flash Memory	33
7.8	USB	34
7.8.1	Board	34
7.8.2	Device	34
7.8.3	Host	37
7.8.4	Both	37
7.9	New Board Design	38
8	Software	41
8.1	YAML Parser	41
8.1.1	Implementation	41
8.1.2	Data Extraction	42
8.1.3	Testing	43
8.1.4	Errors	43
8.2	Remapper	44

8.2.1	Passthrough	44
8.2.2	Configuration	45
9	Evaluation	48
10	Conclusion	51
11	Recommendations	52
12	Reflection	53
13	References	54
	Glossary	57
A	Editorial Modifications	58
A.1	SH	58
A.1.1	Paragraph 1	58
A.1.2	Paragraph 2	58
A.1.3	Paragraph 3	58
A.1.4	Paragraph 4	58
A.1.5	Paragraph 5	58
A.1.6	Paragraph 6	58
A.1.7	Paragraph 7	58
A.1.8	Paragraph 8	59
A.1.9	Paragraph 9	59
A.1.10	Paragraph 10	59
A.1.11	Paragraph 11	59
A.1.12	Paragraph 12	59
A.1.13	Paragraph 13	59
A.1.14	Paragraph 14	59
A.1.15	Paragraph 15	59
A.2	BG	60
A.2.1	Paragraph 1	60
A.2.2	Paragraph 2	60
A.2.3	Paragraph 3	60
A.2.4	Paragraph 4	60
A.2.5	Paragraph 5	60
A.2.6	Paragraph 6	60
A.2.7	Paragraph 7	60
A.2.8	Paragraph 8	61
B	Source Code	62
B.1	Testing	62
B.1.1	git Log	62
B.1.2	Test SWD	62
B.1.3	Test Clock	62
B.1.4	Test LED	62
B.1.5	Test Switch	63
B.1.6	Test Serial	63
B.1.7	Test SPI	64
B.1.8	Test Flash	66
B.2	tinysb	70
B.2.1	git Log	70
B.2.2	hw/bsp/stm32f4/boards/stm32f407usb/	70
B.2.3	hw/bsp/stm32f4/	72
B.3	Remapper	77
B.3.1	git Log	77
B.3.2	src	77
B.4	YAML parser	94

C	Log	97
C.1	Week 1	97
	C.1.1 19/05/25	97
	C.1.2 20/05/25	97
	C.1.3 21/05/25	97
	C.1.4 22/05/25	97
C.2	Week 2	98
	C.2.1 27/05/25	98
	C.2.2 30/05/25	98
C.3	Week 3	98
	C.3.1 02/06/25	98
	C.3.2 06/06/25	98
C.4	Week 4	98
	C.4.1 12/06/25	98
	C.4.2 13/06/25	98
C.5	Week 5	98
	C.5.1 15/06/25	98
	C.5.2 16/06/25	98
	C.5.3 20/06/25	98
C.6	Week 6	99
	C.6.1 24/06/25	99
	C.6.2 25/06/25	99
	C.6.3 26/06/25	99
	C.6.4 27/06/25	99
C.7	Week 7	99
	C.7.1 03/07/25	99
C.8	Week 8	99

List of Figures

1	Keyboard Manager utility showing remappings [1].	9
2	Potential micro-controller candidates.	11
3	Potential flash candidates.	11
4	Scan codes for a QWERTY keyboard.	13
5	Gantt chart at the start of this project.	17
6	Power diagram used for capacitor values [14, p. 78].	18
7	Power pins and capacitors in schematic editor.	19
8	VCAP capacitors.	20
9	Diagram for oscillator capacitors [43, p. 11].	20
10	Schematic section for the oscillator.	21
11	USB ports schematic.	22
12	Voltage regulator.	22
13	SWD pin header in schematic.	22
14	Mode switch.	23
15	SPI schematic.	23
16	Reset button schematic.	24
17	LED schematic.	24
18	Finished circuit schematic.	25
19	PCB design.	27
20	Case design within Onshape.	28
21	Modifications make to the PCB to fix SWD.	31
22	LED turning on during the test program.	32
23	Setup showing serial connection.	33
24	Oscilloscope showing bad USB.	36
25	Modifications make to the PCB to fix USB.	36
26	Erroneous schematic.	38
27	Final PCB design.	39
28	Final board desgin.	40
29	High-level flowchart diagram of YAML configuration file parser and string tokeniser.	41
30	Output from the terminal.	43
31	Working setup with keyboard.	47
32	Gantt chart for this project.	49
33	Full costing excluding time.	50

List of Algorithms

1	Test LED Algorithm	32
2	Test Switch Algorithm	32
3	Test Serial Algorithm	33
4	Test SPI Algorithm	34
5	Test Flash Algorithm	35
6	Passthrough Algorithm	44
7	Remapping Algorithm	45

Listings

1	AutoHotKey script to open notepad when Win+N is pressed.	10
2	Example YAML file.	12
3	Example JSON file.	12
4	Command to flash a <code>stm32f4</code> micro-controller with the <code>test.elf</code> program via <code>openocd</code>	30
5	Log when flashing the PCB fails.	30
6	Log after successfully flashing the PCB.	30
7	Result from SPI Program.	33
8	<code>dmesg</code> logs when failing to connect the PCB.	35
9	<code>dmesg</code> logs when successfully connecting the PCB.	36
10	Example YAML file for configuring the device.	41
11	<code>yaml_file</code> structure definition.	41
12	Top level schema using <code>libcyaml</code> macros.	41
13	Calling <code>cyaml_load_file</code> and handling the return status.	42
14	<code>mapping</code> structure to store keyboard mappings from YAML configuration file.	42
15	Tokenisation of the mappings present in the YAML configuration file.	42
16	Iterator to print mappings to terminal.	43
17	Setting the <code>CYAML_FLAG_OPTIONAL</code>	43
18	<code>malloc</code> function call to allocate memory for storing strings representing the keyboard remappings.	43
19	<code>malloc</code> function calls to allocate memory for the list of string pointers (<code>char **</code>).	44
20	<code>tuh_hid_report_received_cb</code> implementation.	44
21	<code>send_kbd_report</code> implementation.	45
22	<code>send_kbd_report</code> implementation with remappings.	46
23	Log of test code generated via <code>git log</code>	62
24	Code used to test SWD.	62
25	Template code with the clock scalars configured for 168 MHz.	62
26	Test LED Program.	62
27	Test Switch Program.	63
28	Test Serial Program.	63
29	Test SPI Program.	64
30	Test Flash Program	66
31	Log of <code>tinyusb</code> code generated via <code>git log</code>	70
32	<code>tinyusb board.h</code>	70
33	<code>tinyusb family.c</code>	72
34	Log of remapper code generated via <code>git log</code>	77
35	Remapper <code>main.c</code>	77
36	Remapper <code>hid_host.c</code>	78
37	Remapper <code>hid_device.c</code>	82
38	Remapper <code>kbd_remap.h</code>	83
39	Remapper <code>usb_descriptors.c</code>	87
40	Remapper <code>usb_descriptors.h</code>	91
41	Remapper <code>tusb_config.h</code>	91
42	YAML <code>main.c</code>	94

1 Introduction

1 The Computer and Keyboard Agnostic Keyboard Layout (CKAKL) project aims to research, design, and develop a small Universal Serial Bus (USB) device that sits between a users keyboard and computer. ***BG***

2 The device will monitor incoming keyboard events and check these against a list of user configured remappings provided in a configuration file. If the keyboard action has been remapped, the signal will be intercepted and changed to the desired action. ***BG***

3 This device aims to enable the user to move their desired remappings between keyboards and computers without the need for any expensive peripherals or software on the host machine. Additionally, this will improve the users productivity and efficiency as no additional time is needed in configuring each individual setup, which is particularly prolific in areas such as universities and workplaces operating hot desks. ***BG***

4 This project will contain research on products currently available on the market that perform the same actions and identify any shortfalls within them. A prototype device will then be designed using KiCAD and procured from a Printed Circuit Board (PCB) manufacturer. Software will be developed to run on a micro-controller that will be responsible for remapping the keyboard events, as well as providing a configuration mode so the user can define the remappings. Finally, testing will be conducted to ensure the device satisfies all of the essential requirements outlined later in this report. ***BG***

5 This report will contain sections relating to the different research and development stages, these include:

- Literature Review
- Hardware Research
- Software Research
- Requirements
- Project Management
- Hardware Design
- Hardware Testing
- Software Design
- Evaluation
- Conclusion
- Recommendations
- Reflection

BG

2 Literature Review

1 Remapping a keyboard via software has long been possible on major operating systems: Windows with the Keyboard Manager utility [1] as part of their PowerToys suite, Linux - requiring Xorg - having the X keyboard extension [2], and for macOS the `hidutil` tool [3]. These can all configure a sequence of keyboard inputs to be replaced by another. The Keyboard Manager configuration tool can be seen in Figure 1. The problem with the Windows solution is that the PowerToys suite does not come pre-installed with Windows, and the user may not be able to install it for various reasons. The problem with the Linux and macOS solutions are that they require a high level of understanding of the system being worked on. **JG**

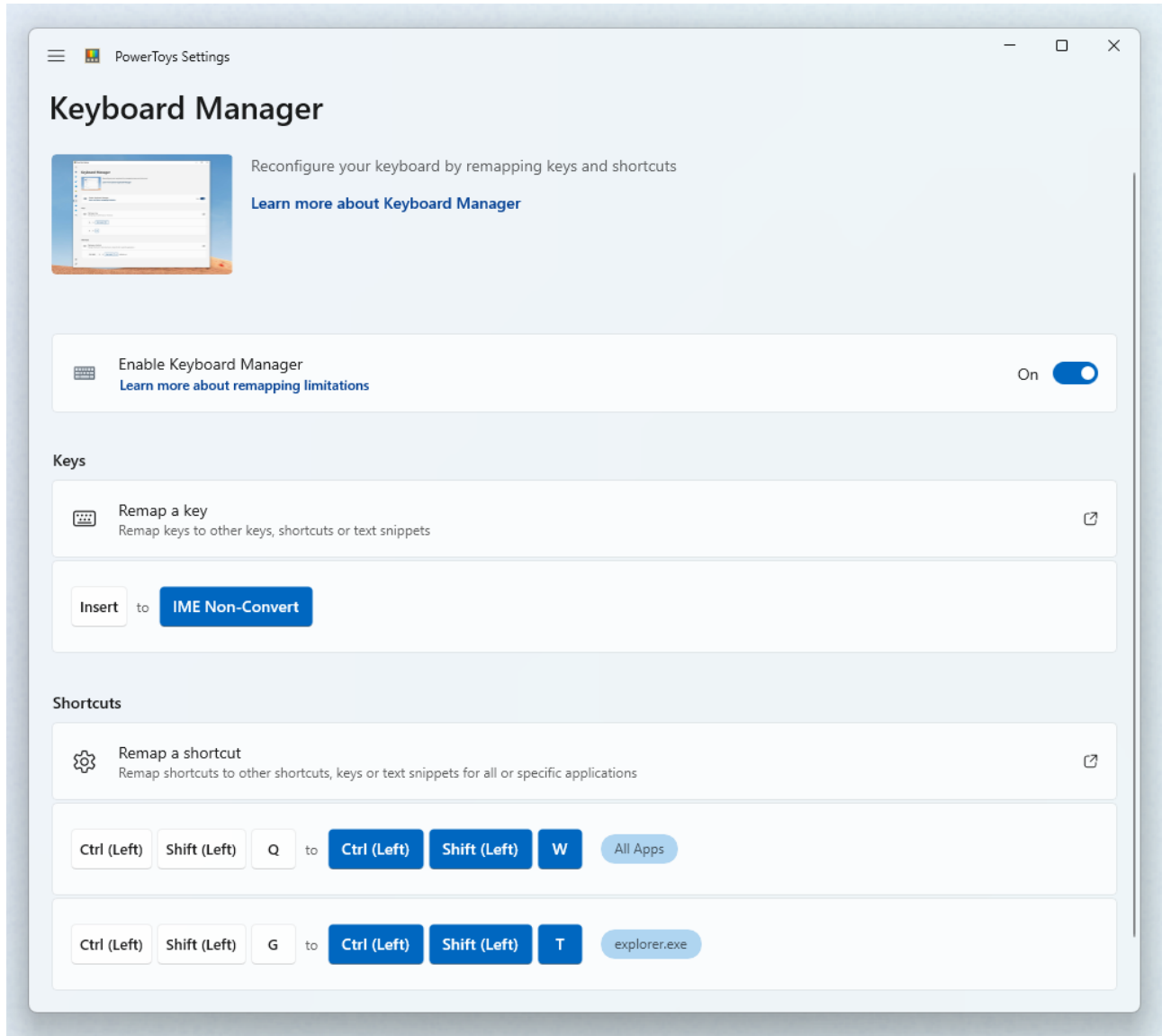


Figure 1: Keyboard Manager utility showing remappings [1].

2 There are a few hardware keyboard remappers [4] [5] [6], the most complete being the HID Remapper [7]. It allows the user to fully remap their keyboard, mouse, and controllers via a web interface and Python script. A Raspberry Pi Pico intercepts the peripherals inputs and modifies them according to the configuration before they are sent to the computer. In addition to remapping keys, it has support for macros as well as analogue input methods. **JG**

3 A common tool used on Windows to handle more complex remappings, such as macros and scripting, is AutoHotKey [8]. It has a custom syntax to describe tasks and actions that can then be bound to a keyboard. An example of this keybinding is shown in Listing 1, which opens notepad when `Win+N` is pressed. **JG**

```
1 #n::Run "notepad"
```

Listing 1: AutoHotKey script to open notepad when **Win+N** is pressed.

4 There is currently an appetite for such a product within the consumer market in the form of a platform agnostic tool due to the volume of software in existence [9], likely used by technicians who frequently move between workstations. This is compounded by negating the need for software to be downloaded, as such, may not be suitable for environments where internet connections or extra downloaded content isn't permissible. With the advent of more advanced games and simulation environments, solutions to adapting more advanced mechanisms are required to input commands [10]. This may require the need to completely change the inputs parsed due to a lack of software support. **SH**

3 Research

3.1 Micro-controller

1 The processing power for the project will be provided by a micro-controller. The primary requirements for this project are two USB controllers, small package, and low power consumption. Research indicated that the maximum amperage was 500 mA in the USB 2.0 standard [11]. As such, a lower power chip was required to safely power the keyboard. **SH** [JG] $\leftrightarrow$

2 The requirement for interacting with both a computer and keyboard results in the need to be a device and host concurrently. Alternatively, two separate chips could be used which communicate and perform the host and device roles separately. The table in Figure 2 lists some of the key criteria evaluated from the chips researched. **SH** [JG] $\leftrightarrow$

Name	USB Support	Current consumption (max) [mA]	Package	Notes
STM32F1XX	USB FS $\times 1$	50 (150)	BGA64	[12]
STM32F3XX	USB LS $\times 1$	75 (100)	WLCSP100	[13]
STM32F4XX	USB HS $\times 1$ & USB FS $\times 1$	117 (240)	WLCSP90	[14]
PIC16F877A	N/A	15 (240)	QFN28	[15]
PIC18F2550 & PIC18F4550	USB 2.0 FS $\times 1$	70 (250)	QFM44	[16] [17]
ATmega16U4 & ATmega32U4	USB 2.0 FS $\times 1$	15 (27)	QFN44	[18]
FT4232H	USB HS $\times 2$	70 (150)	QFN64	[19]

Figure 2: Potential micro-controller candidates.

3 There are a limited amount of chips that support simultaneous USB instances. The most likely family of chips that could be used is the STM32F4XX or FT4232H. They both support two USB interfaces concurrently, which allows both the keyboard and computer to interact with minimal latency. The purpose of the FT4232H chip is to convert USB messages into Universal Asynchronous Receiver/Transmitter (UART) or Inter-Integrated Circuit (I2C) messages with no further processing and cannot be programmed, making it unsuitable. Alternatively, the STM32F407VGT6 fulfils the use requirements and has an adequately small package. Another factor that benefits the development using the STM32F407VGT6 is access to development boards that can be used out of the box. This will greatly aid in the creation of software and prototyping concepts for the hardware before a PCB is created and manufactured. **SH** [JG] $\leftrightarrow$

3.2 Flash Memory

1 Due to the proposed method of storing the configuration files, an external storage device is required. The likely configuration file size would be less than 64 Mbit hence, an apt size would be 64 Mbit to 256 Mbit. A smaller package would be ideal due to the size constraints in place. Serial Peripheral Interface (SPI) flash would be ideal due to the limited space to run several traces between circuits and inbuilt support with most micro-controllers. Figure 3 details the specifications of various flash memory chips. **SH**

Name	Storage size [Mbit]	Current consumption (max) [mA]	Package	Notes
S25FL256SAGNFI001	256	16 (33)	WSON	[20]
S25FL256LAGNFI010	256	10 (30)	SOIC 8	[21]
S25HS512TDPBHI010	512	10 (53)	24-Ball BGA	[22]
W25Q32JVSSIQ	32	20 (25)	VSOP 208	[23]
W25Q128JVSIQ	128	20 (25)	SOIC 208	[24]

Figure 3: Potential flash candidates.

2 The minimum storage size for an FAT16 file system is 128Mbits. Hence, a large chunk of the pool can be omitted. Another factor is the size of the chips, along with the connections required. As such, a likely suitor for the PCB will be the W25Q128JVSQ. It has a relatively low power draw and a reasonable package size, meaning that it can fit into the expected form factor easily. **SH**

3.3 Configuration File

1 A configuration file will be present on the device to enable the user to customise the device identifier (DID), vendor identifier (VID), keyboard remappings, macros, and multimedia keys. To achieve this either Yet Another Markup Language (YAML) or JavaScript Object Notation (JSON) can be used. **BG [JG]** 

2 YAML is a light-weight, human readable data representation language. Primarily, it is designed to be easy to read and understand whilst still providing functionality for complex features [25]. An example YAML file is shown in Listing 2. **BG [JG]** 

```
1 Origin:
2   author: Dan Brown
3   language: English
4   publication-date: 2017-10-03
5   pages: 461
6   description: | When billionaire researcher Edmond Kirsch is killed,
7                 it is up to Robert Langdon & Ambra Vidal to honor
8                 his memory by making public his findings concerning the
9                 origin of human life and its destiny.
```

Listing 2: Example YAML file.

3 JSON is a human readable language which is independent from the programming language being used for the application. It is most commonly used in web-based applications. An example JSON file is shown in Listing 3. **BG [JG]** 

```
1 {
2   "Origin": {
3     "author": "Dan Brown",
4     "language": "English",
5     "publication-date": "2017-10-03",
6     "pages": 461,
7     "description": "When billionaire researcher Edmond Kirsch is killed,
8                   it is up to Robert Langdon & Ambra Vidal to honor
9                   his memory by making public his findings concerning
10                  the origin of human life and its destiny."
11   }
12 }
```

Listing 3: Example JSON file.

4 For this device, a YAML configuration file will be used due to it requiring less syntax, making it easier for the user to understand, as well as reducing errors from incorrect syntax. Additionally, comments can be present in YAML files allowing for extra information to be present to aid the user in configuring the device. **BG [JG]** 

3.4 Keyboard Features

1 On the market there are a wide range of keyboards which all come with varying features. The majority of these features can be put into one of the following categories:

- Remapping of keys which is the process of changing what action or character is associated with a specific key on the keyboard [26].
- Keyboard macros which allow for a sequence of key presses or computer actions to be mapped to single key press [27].
- Multimedia keys which are typically present on the majority of keyboards to enable easy control of volume and media playback such as playing and pausing a video [28].

4 Requirements

1 The device must meet the following essential requirements:

- Not require the user of device drivers or installation of specific software on the host machine.
- Be user configurable via the use of a YAML configuration file present on the device.
- Able to mimic a keyboard DID and VID if specified in the YAML configuration file.
- Able to sit in series of a keyboard and host machine and not effect keyboard functionality.
- Have a physical switch to alternate between active and configuration modes.
- Support remapping of keys and multimedia keys.
- Provide USB 2.0 compatibility.
- Use a file system that is widely supported by a range of operating systems.
- Contain and display error light sequences to notify the user of particular errors.
- Comply with Electronic Equipment (safety) regulations 2016 [30].
- Comply with Restriction of Hazardous Substances (RoHS) regulations [31].
- Comply with waste electrical and electronic equipment (WEEE) regulations 2013 [32].
- Comply with the Electromagnetic Compatibility regulations 2016 [33].

BG

2 The device may contain functionality to meet the following desirable requirements:

- Packaged in a small form factor.
- Able to detect and give the user verbose error logging.
- Minimal added latency to keyboard inputs.
- Provide a tool with a graphical user interface to generate the YAML configuration file.
- Provide support for all keyboard macros.

BG

5 Project Management

5.1 Methodology

1 This project will utilise the Rapid Application Development (RAD) development methodology. RAD enables the project team to focus on quick and iterative release cycles, primarily focused on delivering a working product in a shorter timeline. **BG**

2 RAD is advantageous to the project due to it being designed to help decrease the development time for the project as well as, being adaptable to any changes there might be in the requirements throughout the lifetime of the project. Moreover, the progress of the project can be measured accurately due to the development stages being broken down into small stages. **BG**

3 The major disadvantage of the RAD methodology is any part, or sequence of parts, of the project that cannot be parallelised will increase the overall development time [34]. **BG**

4 For this methodology to work, the project has been split into the major tasks which in turn are broken up into smaller sub-tasks:

- Software Development
 - USB Stack to Host
 - USB Stack from Keyboard
 - YAML File Parser
- Hardware Development
 - Schematic Design
 - PCB Design
 - PCB Procurement
- Integration
 - Hardware Testing
 - Software Deployment
 - Software and Hardware Testing

BG [JG] 

5.2 Ethics

1 Due to the very limited data processing this device is performing, there are only three main ethical impacts to be considered: mimicking device and vendor IDs; PCB manufacturing; and WEEE legislation. **BG**

2 The action of mimicking device and vendor IDs of other products can fall under the Computer Misuse Act 1990 [35]. If this is done without the users knowledge and consent it would be deemed as a form of malware as the device is pretending to be something that it is not. To be compliant with this legislation, inside of the YAML configuration file there will be an option to specify the device and vendor ID of the keyboard device the user wishes to mimic. If no device and vendor ID is specified, the device will default to “Generic USB Device”. **BG**

3 PCB manufacturing has a variety of ethical considerations such as environmental impacts, worker safety and fair labour practices [36]. For this device, the manufacturer JLCPCB was chosen to produce the PCB. Outlined on the website of the manufacturer are certificates of RoHS compliance as well as various International Organization for Standardization (ISO) certificates [37]. These certifications provide confidence that the environmental impacts are limited to the required standard and substances such as lead, which are harmful to workers, are not used during the manufacturing process. **BG**

4 However, there is no information regarding labour practices for the chosen manufacturer which is common across most PCB manufacturers. JLCPCB manufacturing factories are based in China so the labour practices cannot be accounted for. The device could have been manufactured with a UK based company such as Newbury Electronics [38], meaning labour practices would need to be compliant with UK regulations. Due to Newbury Electronics being located in the UK, the cost of having PCBs manufactured and assembled is significantly higher when compared to JLCPCB and you may be able to infer the labour practices based on this vast difference. The project decided to use JLCPCB as it appeared to be the best value for money when considering the cost of the PCB manufacturing and assembly process as well as the ethical implications associated with both manufacturers. **BG**

5 WEEE legislation [32] applies for the disposal of the product at the end of its life. In the UK, the main responsibilities of the manufacturer and retailer are to provide the customer with information as to how to recycle the electronic device appropriately and offer a free take-back of the device. Additionally, a crossed out wheeled bin symbol must be present on the device to ensure conformity to BSI EN50419. As this symbol would need to be present on the chassis (enclosure) of the device, this is out of scope for the project. **BG**

5.3 Gantt Chart

1 Figure 5 shows the initial Gantt chart for the project. **SH**

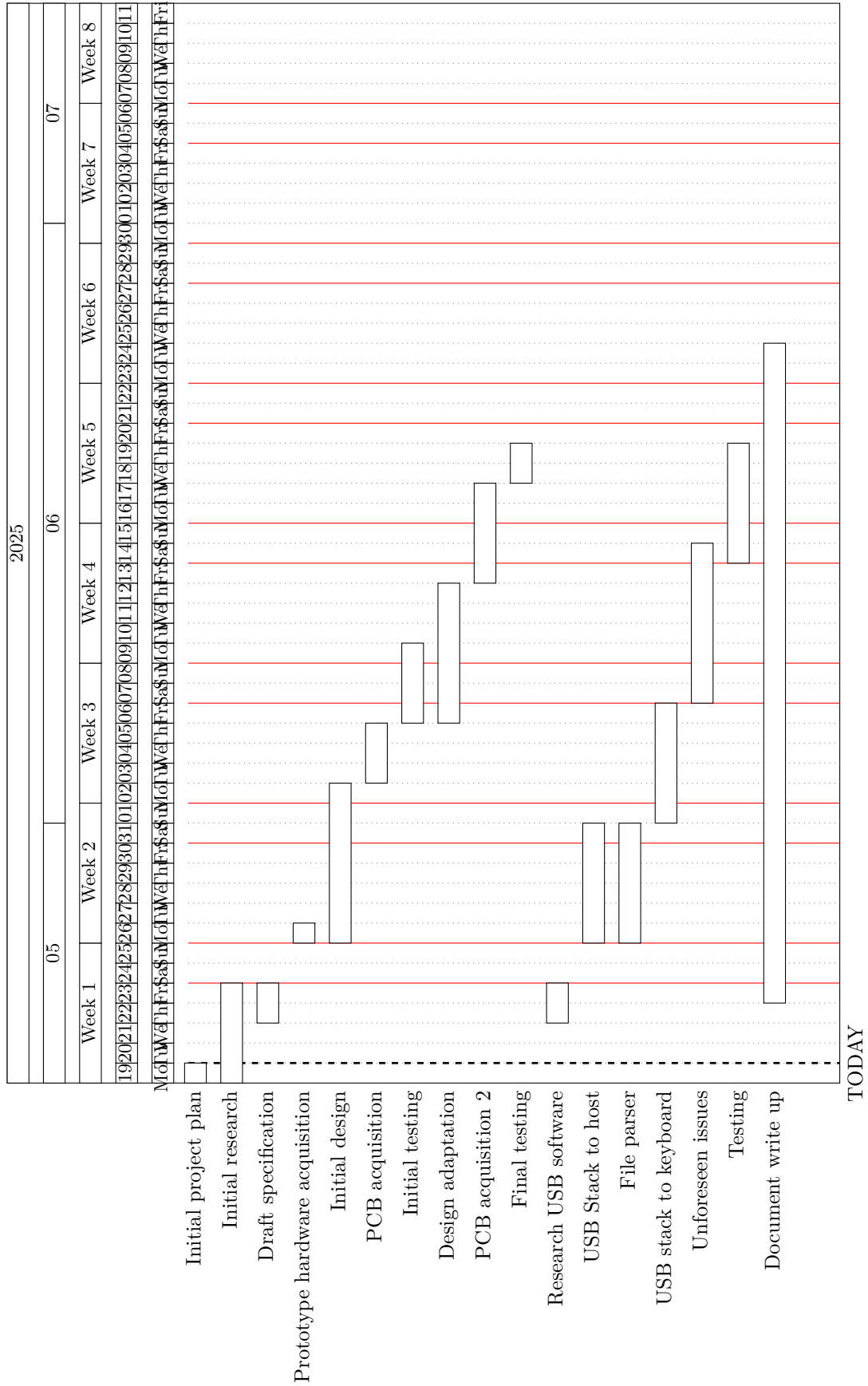


Figure 5: Gantt chart at the start of this project.

1 The KiCAD electronics design suite [39], an open source computer-aided design (CAD) program, was used to create both the schematic and PCB. Some custom footprints and symbols were required to accommodate parts not within the internal catalogue [40] [41] [42]. ***SH*** [***JG***]

2 A draft bill of materials was created to break down the schematic into sizeable chunks:

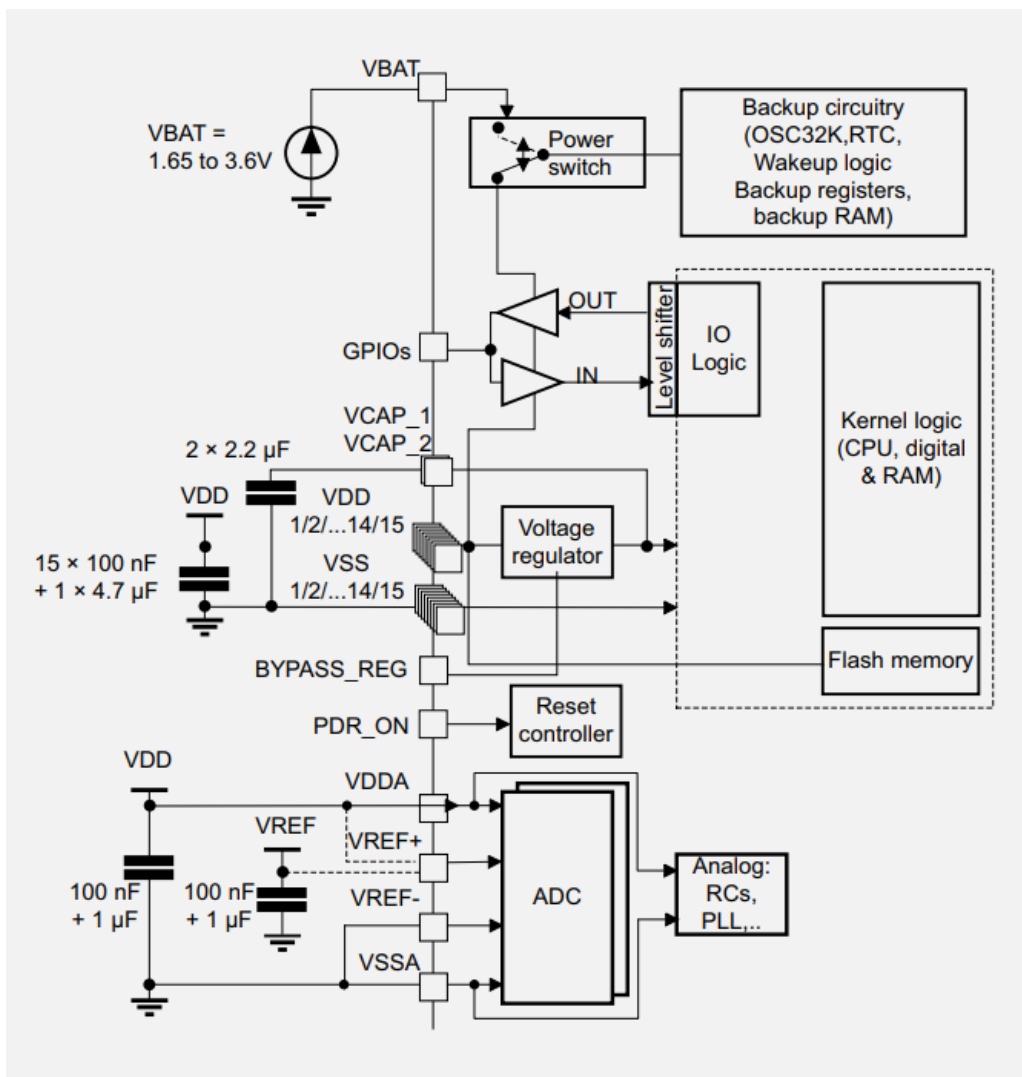
Micro-controller	STM32F407VGT6
Voltage regulator	LMS8117AMP-3.3/NOPB
Flash storage	W25Q128JVSIG
Oscillator	X50328MSB2GI

 SH

6.1 Micro-controller

1 The STM32F407VGT6 specification [14, pp. 38, 78] was used as a reference for the features that would be implemented into the design. ***SH***

2 The first component added to the schematic was the STM32F407VGT6, followed by connecting its power and ground pins - with 100 nF decoupling capacitors - according to the specification [14, p. 43]. 100 nF decoupling capacitors were added to ensure that power and ground were adequately smoothed, shown in Figure 6. The reference voltage for the logic, VDDA, required a larger 1 μ F capacitor. Figure 7 shows the power and ground pins connections. **SH** **[JG]** $\hookrightarrow$



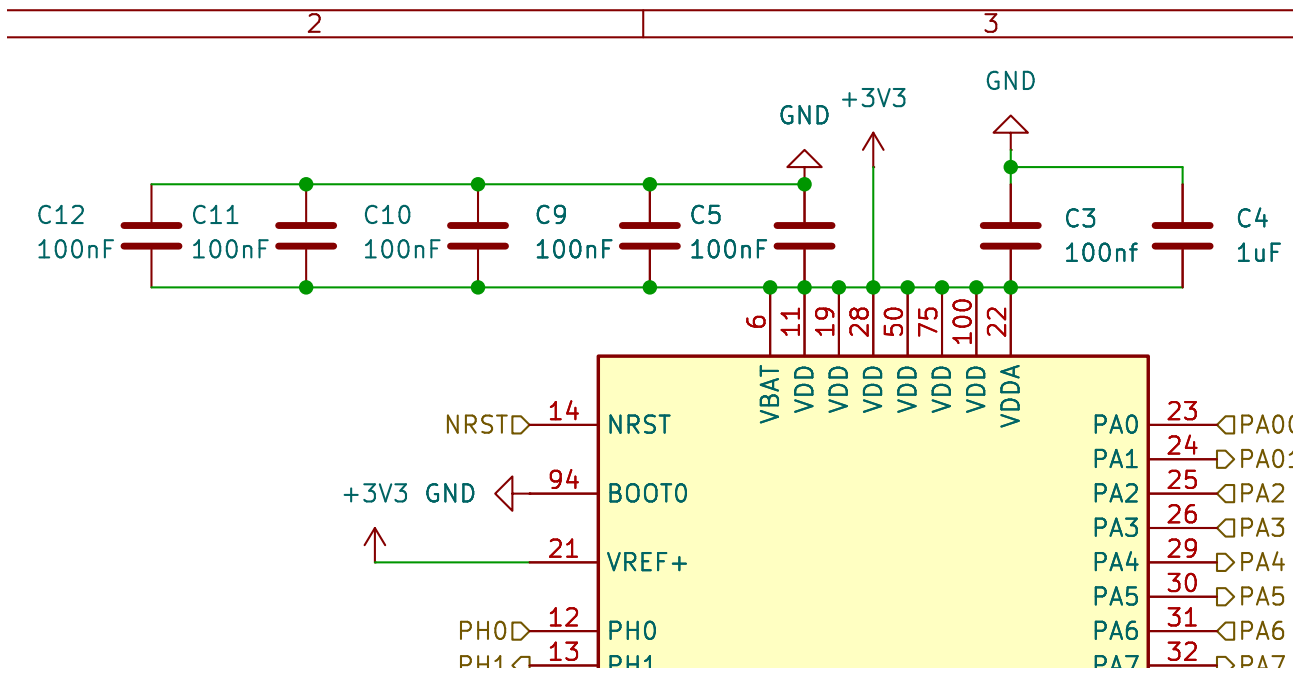


Figure 7: Power pins and capacitors in schematic editor.

6.1.1 Review

1 When the schematic was reviewed by an experienced design engineer, it was recommended to add 2.2 μF capacitors for the VCAP pins, as shown in Figure 8. *SH* [JG] 

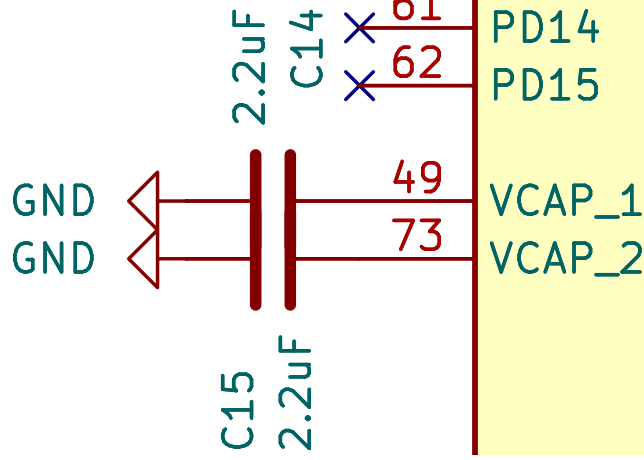


Figure 8: VCAP capacitors.

6.2 Oscillator

1 For FS USB to be accessible, an external oscillator was added through following AN2867 [43]. To ensure a wide range of frequencies is achievable, an 8 MHz oscillator was chosen. An internal prescaler can be used to achieve any of the frequencies required. The oscillator was connected to the dedicated external oscillator pins PH0 and PH1. *SH* [JG] 

2 The main equation used for calculating the decoupling capacitors values [43, p. 12] and the associated diagram in Figure 9.

$$C_L = \frac{C_{L1} \times C_{L2}}{C_{L1} + C_{L2}} + C_s \quad (1)$$

Given that $C_s = 5$ [44] and $C_L < 20$ pF, $C_{L1} = C_{L2} = 20$ pF. Figure 10, shows the oscillator design. *SH* [JG] 

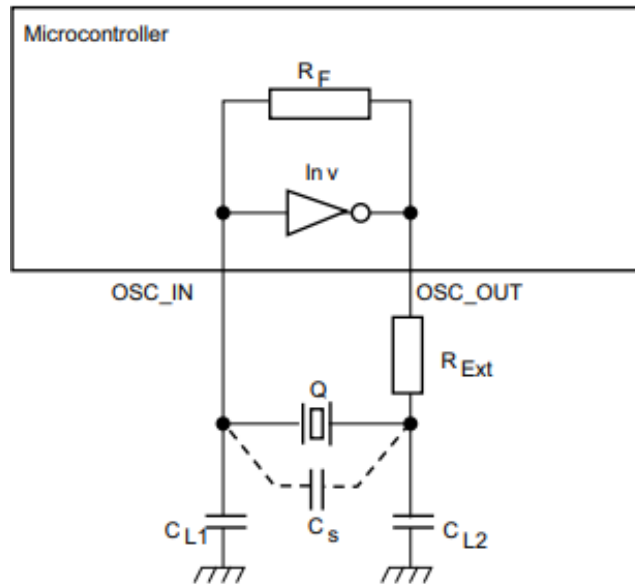


Figure 9: Diagram for oscillator capacitors [43, p. 11].

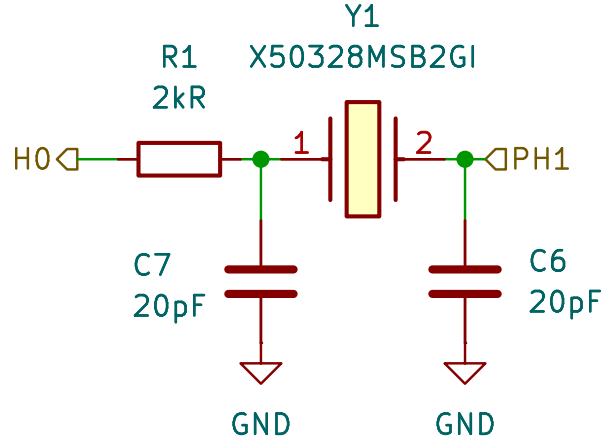


Figure 10: Schematic section for the oscillator.

6.3 USB Connection

1 The overarching information used for designing the USB ports was found in the USB 2.0 specification [45]. USB Type A connectors were used due to their compatibility with most keyboards. **SH**

6.3.1 Device

1 The micro-controller connects to its host, e.g. a laptop, through the USB HS interface. The data lines D+ and D- from this connector go to pins PA12 and PA11 respectively. 5 V and ground, via this connector, supply power to both the board and downstream devices. **SH [JG]**

6.3.2 Host

1 The micro-controller connects to a device, e.g. a keyboard, through the USB FS interface. The data lines D+ and D- from this connector go to pins PB15 and PB14 respectively. 5 V and ground power pass-through was attached to. **SH [JG]**

2 To negate over-current, a current limiting switch was added. It can be enabled using pin PA1 and will output a signal - if the switch detects too much current going to the device - via pin PA0. TPS2051BDBVR was chosen due to its compliance with the USB limits and small package [46]. **SH**

6.3.3 Review

1 When the schematic was reviewed by an experienced design engineer, it was noted that there were no pull-up or pull-down resistors for the host and device data lines making the connectors inoperable and non-compliant. As specified [45], the D+ line of the device connector required a 15 k Ω pull-up resistor and the D+ and D- lines of host connector each required 15 k Ω pull-down resistors. Figure 11 shows the schematic for both host and device components. **SH [JG]**

6.4 Voltage regulator

1 The 5 V input from the device USB connector is too high for most of the components. As such a voltage regulator is required to ensure a 3.3 V source. Figure 12 shows the connections and the 10 μ F capacitors recommended by its specification [47]. **SH**

6.5 Programming Interface

1 To program the micro-controller, either Joint Test Action Group (JTAG) or Serial Wire Debug (SWD) is required. A SWD pin header was chosen, due to a smaller connector being required. 3.3 V, SWCLK (PA14), ground, and SWDIO (PB3) were connected to pins 1 to 4 respectively, shown in Figure 13. **SH [JG]**

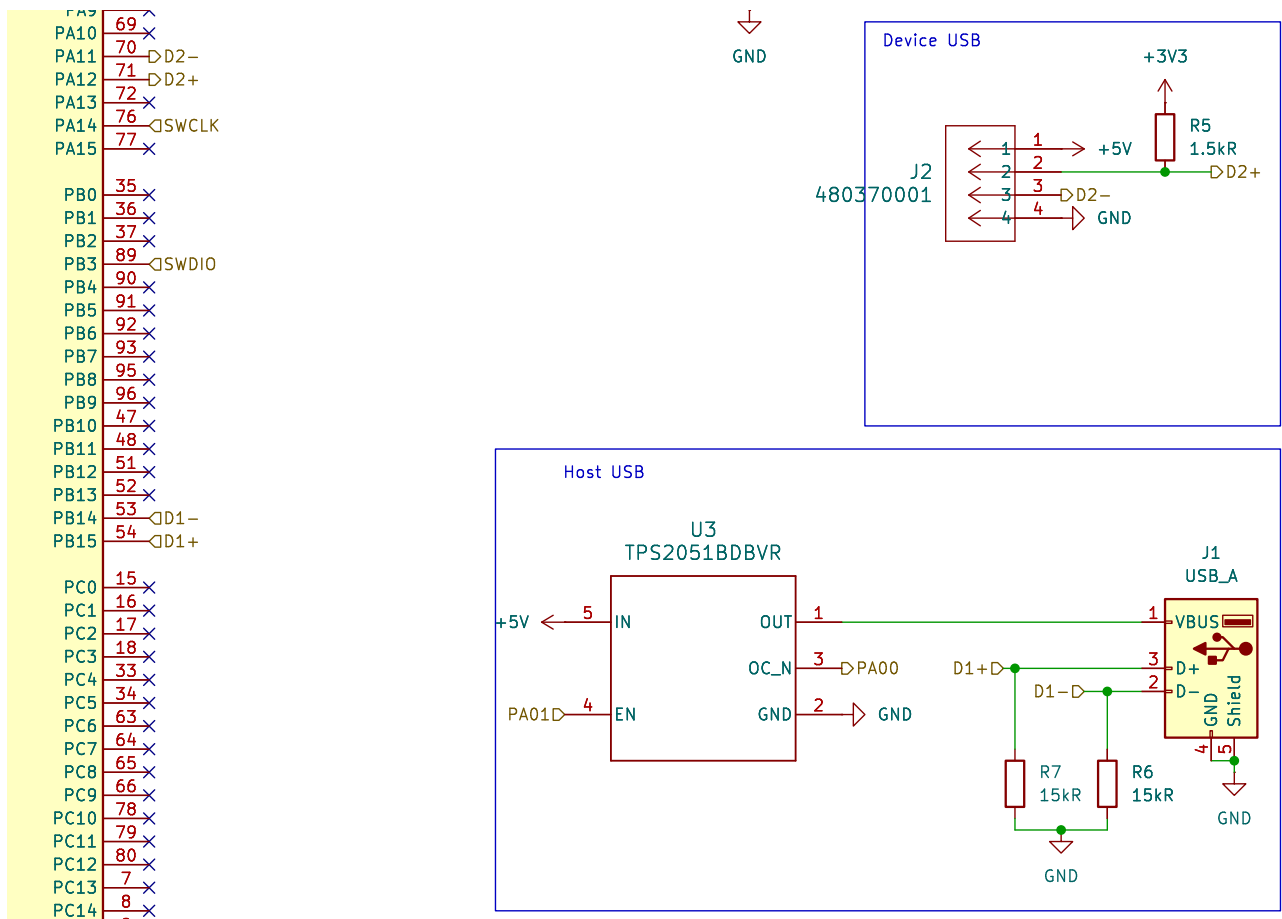


Figure 11: USB ports schematic.

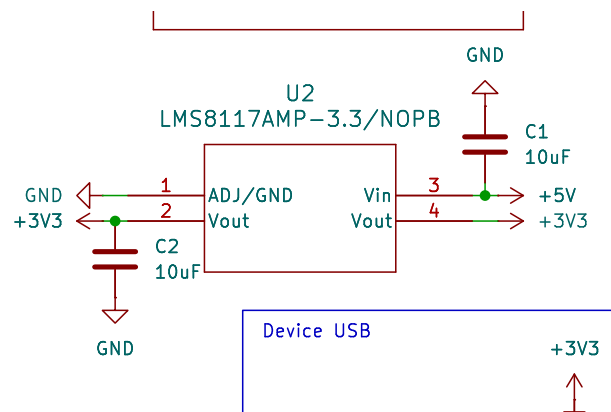


Figure 12: Voltage regulator.

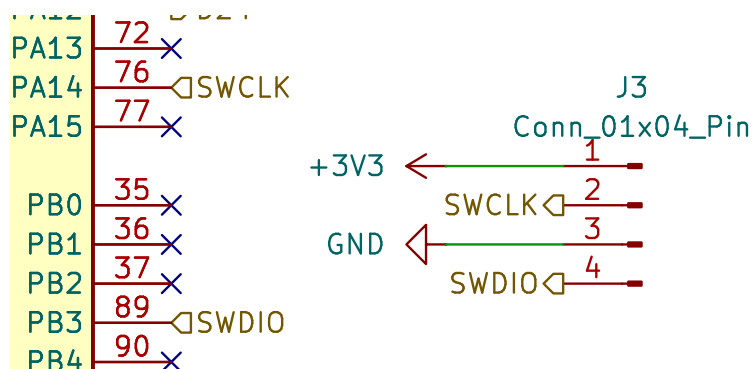


Figure 13: SWD pin header in schematic.

6.6 Mode Switch

1 A switch is required to change modes, so that the ability to upload YAML files is toggleable. The switch decided upon was the SSSS925800 [48] due to its small package. In conjunction with this, a low pass filter was added to ensure that bouncing will not occur during transition between states. This was achieved through modelling the filter characteristics and the attenuation of common bouncing frequencies. The full mathematical model can be found in the appendices. This was then tested in Falstad [49], to ensure accuracy. $R = 33$ and $C = 4.7e - 6$ was decided upon to balance the cutoff point and sharp fall off. This is shown in Figure 14. **SH** [JG] $\boxed{\hookrightarrow}$

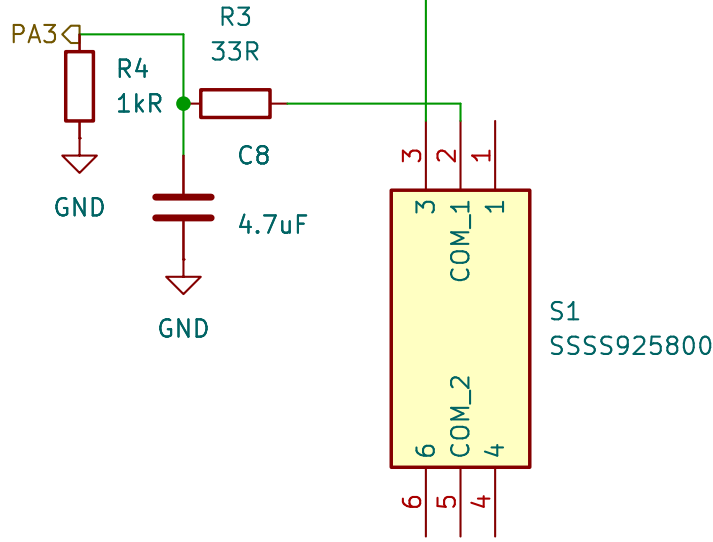


Figure 14: Mode switch.

6.7 Flash Memory

1 From the research given in Section 3.2, the chip used will be the Winbon W25Q128JVSQ [24]. This was used to attach CS, CLK, MISO, and MOSI to the micro-controller pins for the internal SPI1 interface PA4–7 respectively. **SH**

2 The flash is connected via SPI to the micro-controller using pins PA4 (CS), PA5 (CLK), PA6 (MISO), and PA7 (MOSI). These pins connect to the SPI1 interface - a hardware module on the micro-controller for handling SPI, as shown in Figure 15. **JG**

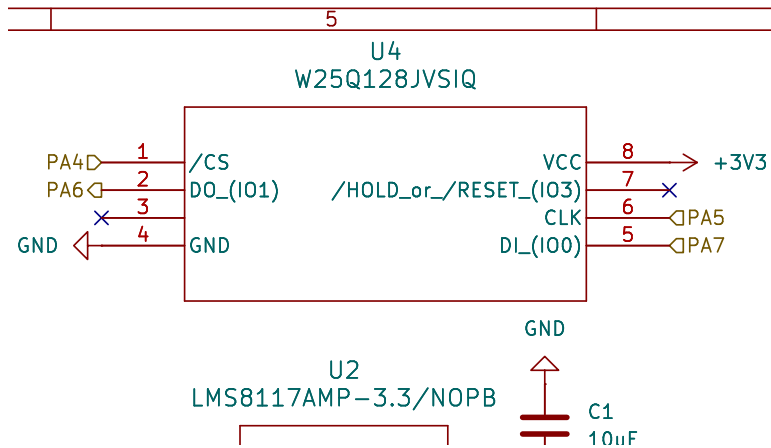


Figure 15: SPI schematic.

6.8 Reset Button

1 A reset button was required to aid the quick development and debugging of software. This allows resetting the chip without disconnecting the power. This was achieved through a button, with an extra capacitor to debounce the signal, pulling down the active high signal on the NRST pin of the micro-controller (Figure 16).

SH [JG] 

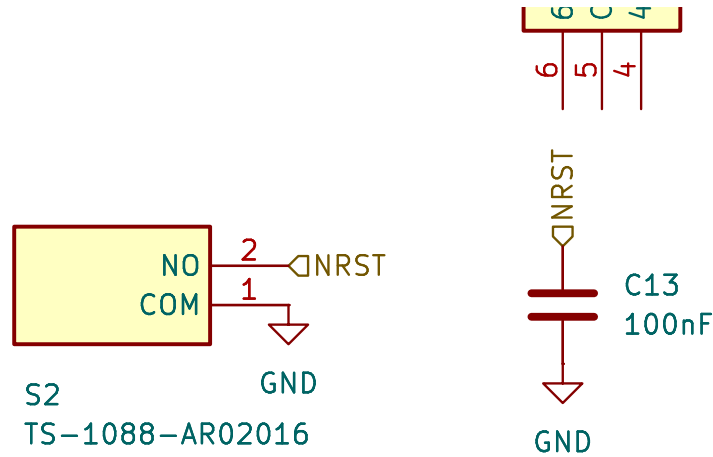


Figure 16: Reset button schematic.

6.9 LED indicator

1 A light emitting diode (LED) indicator was required for clear indication of errors and feedback to the user. This utilised pin PA2 being able to sink current and use its internal pull-down (Figure 17). This was connected to the 3.3 V power rail through a resistor whose values was calculated using $R = (V_s - V_f)/I_f$. Therefore, for $V_s = 3.3\text{ V}$, $V_f = 1.875\text{ V}$, and $I_f = 25\text{ mA}$, $R = 75$. SH

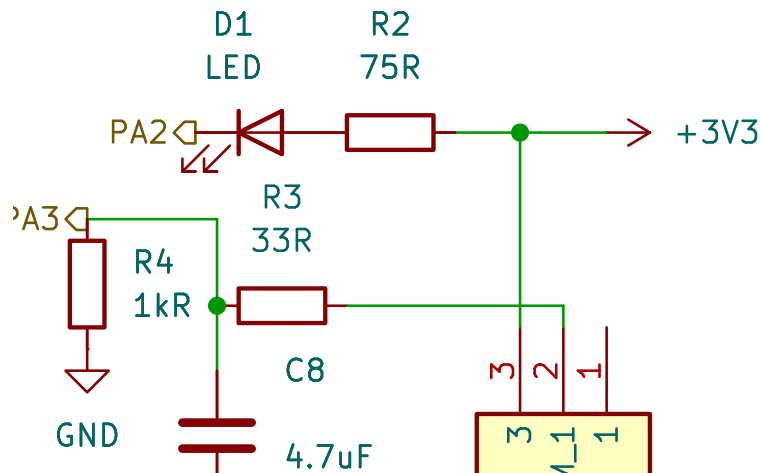


Figure 17: LED schematic.

6.10 Finished Schematic

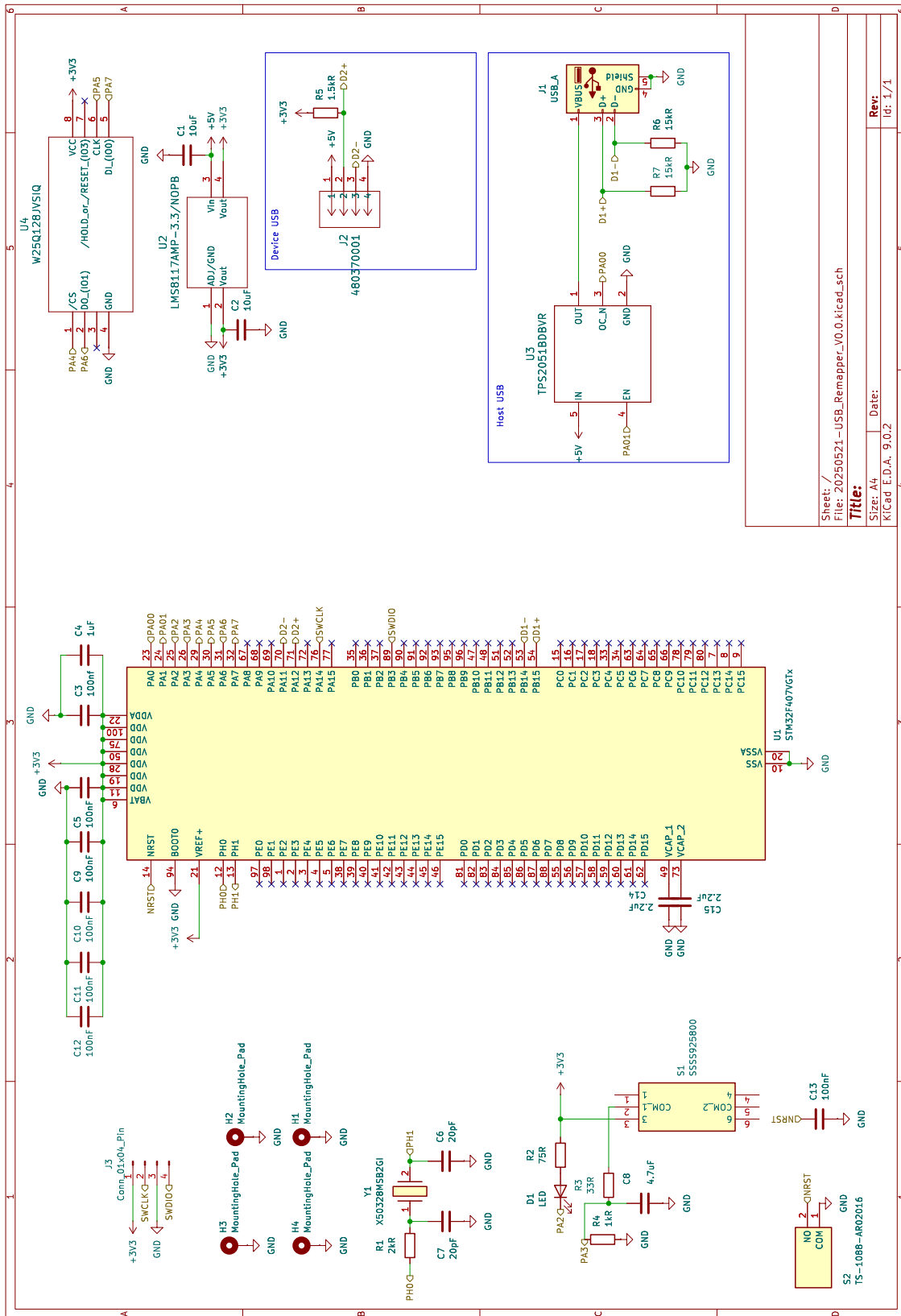


Figure 18: Finished circuit schematic.

6.11 PCB Design

1 The netlist created from the schematic was imported into the PCB editor along with the required footprints. The components were placed to ensure that the thinnest width was possible. Components were added in the order that they would be connected. Hence, the SWD, current limiting switch, and SPI were near the top of the board. The USB data lines were connected via a differential pair, which was length matched to ensure a similar impedance. The final design can be seen in Figure 19. *SH* [*JG*] 

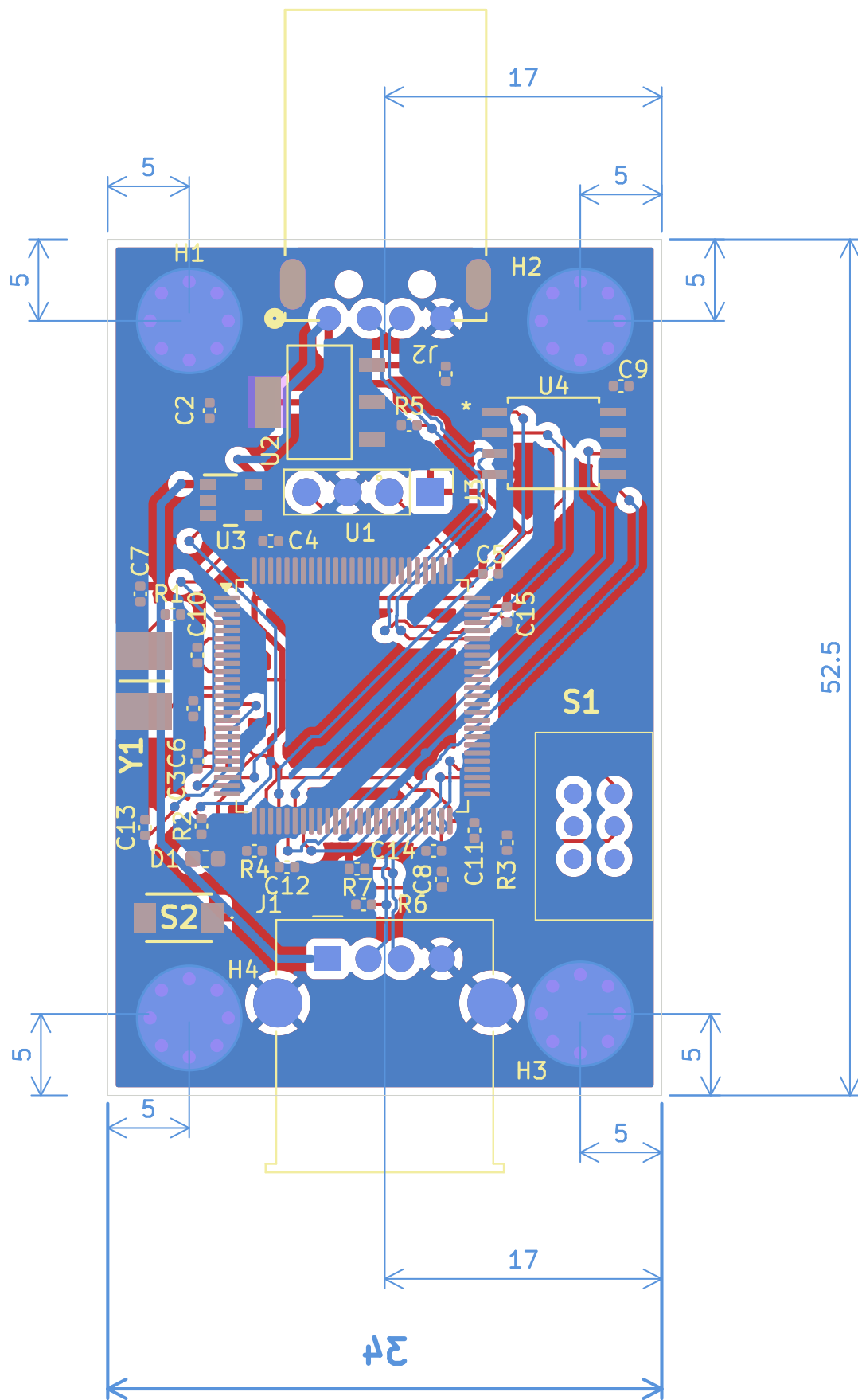


Figure 19: PCB design.

6.12 Production

1 The board was acquired through JLCPCB [50], who produced and partly assembled the board. JLCPCB was chosen due to their short lead times and fast turnaround, which greatly increases the amount of time that can be used for troubleshooting and software development. To export the design, a plugin for KiCAD was used [51], which exports the bill of materials, positions and Gerber files. To finish the production process, all through hole components were added along with the voltage regulator. These were added in accordance with company electronics and soldering guidance specified in JSP482 [52]. **SH**

6.13 Case Design

1 Onshape [53] was used for the case design. This is due to it being a free service, which outputs Standard Tessellation Language (STL) files that most 3 dimension (3D) printer slicers use. The case was designed with an open top, and pins to hold the PCB in place. Figure 20 shows the case within Onshape. Chitubox [54], a slicer for resin printers was used to convert the STL file into a proprietary file format used by the printer. An Anycubic Photon was used to print the cases [55]. **SH**

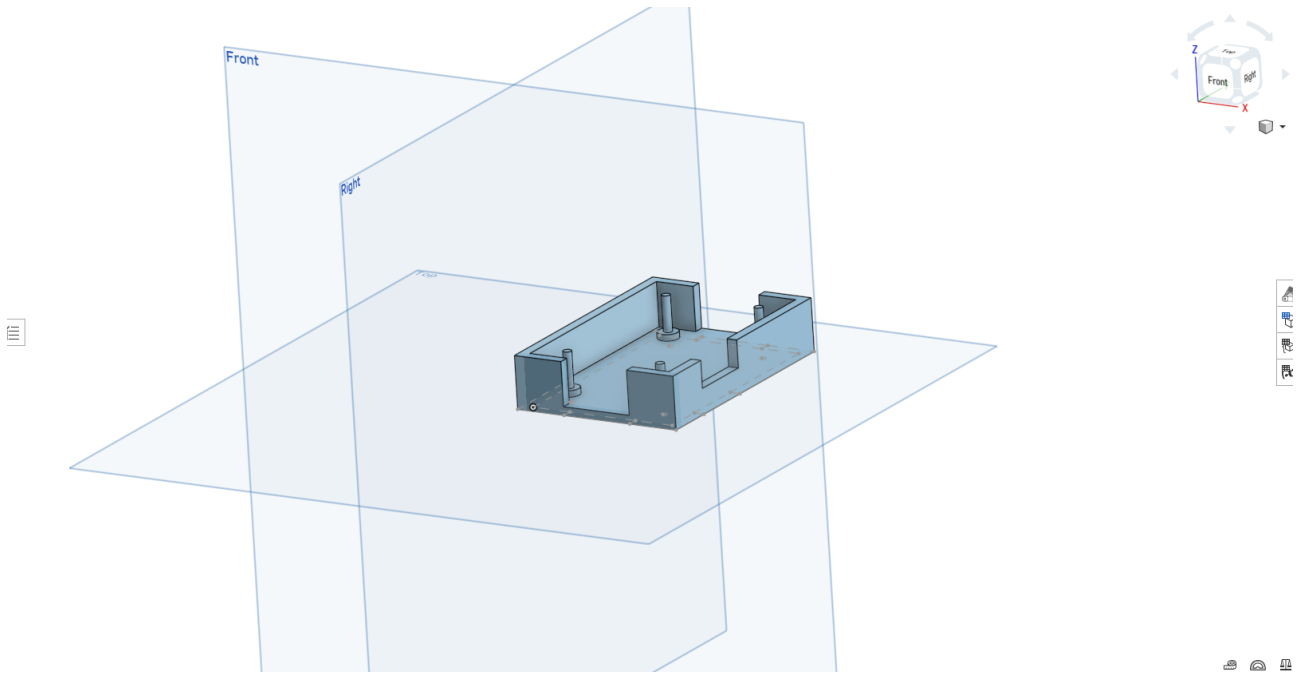


Figure 20: Case design within Onshape.

6.14 Pinout

1 Table 1 shown below represents the current pinout for the board. It contains some of the necessary information that directs the software development. **SH**

Pin	Purpose	Notes
VDD, VBAT & VDDA	3.3 V power supply	Connects to voltage regulator (U2), decoupling capacitors. VDDA is used for logic.
PA0	Overcurrent	Connects to current limiting switch (U3), used to create feedback loop.
PA1	Enable VBUS	Connects to current limiting switch (U3), active high.
PA2	LED output	Connected to LED (D1), active low.
PA3	Mode switch input	Connects to switch (S1) & R3.
PA4	SPI Chip select	Outputs to SPI Flash Storage (U4), active low.
PA5	SPI Clock signal	Outputs to SPI Flash Storage (U4).
PA6	MISO	Outputs to SPI Flash Storage (U4).
PA7	MOSI	Input from SPI Flash Storage (U4).
PA11	Device USB –	Connects to male USB connector (J2).
PA12	Device USB +	Connects to male USB connector (J2).
PA14	SWCLK	Input from SWD Pin connector (J3).
PB3	SWDIO	Input from SWD Pin connector (J3).
PB14	Host USB –	Connects to female USB connector (J1).
PB15	Host USB +	Connects to female USB connector (J1).
PH0	Oscillator input	Connects to oscillator (Y1).
PH1	Oscillator output	Connects to oscillator (Y1).
NRST	Reset input	Connects to reset button (S2).

7 Hardware Testing

1 With the PCBs assembled, it was time to test them. To verify the hardware was working correctly, programs were written to individually test each component of the PCB. *JG*

7.1 Power On

1 Initial power on testing was conducted via pins 1 and 3 of the SWD header. This helped catch any errors within the circuit and prevent overloading the USB power supplies of any upstream hosts. This verified there were no shorts on the 3.3 V net. The voltage regulator was also found to be operating correctly using the 5 V input from the USB connector. As a result, the hardware is less likely to be damaged by overcurrent or overvoltage. *SH*

7.2 SWD

1 The SWD protocol is used to debug micro-controllers via a debugger which in turn is controlled by a computer. Open On-Chip Debugger (OpenOCD) [56] is used to control the debugger from the computer and can flash the micro-controller via the command in Listing 4. *JG*

```
1 #!/bin/sh
2 openocd -f interface/stlink.cfg -f target/stm32f4x.cfg -c "program test.elf verify
  reset"
```

Listing 4: Command to flash a `stm32f4` micro-controller with the `test.elf` program via `openocd`.

2 The program in Listing 24 was used to verify that the tooling on the computer was configured correctly to build programs targeting the micro-controller, and that a program could be flashed and executed on the micro-controller. *JG*

3 However, when flashing the PCB, it failed with the error shown in Listing 5. At first it was unclear if the `openocd` command was incorrect, but after verifying it working on a development board it became clear that there was a problem with the PCB. *JG*

```
1 Open On-Chip Debugger 0.12.0
2 Licensed under GNU GPL v2
3 For bug reports, read
4   http://openocd.org/doc/doxygen/bugs.html
5 none separate
6
7 Info : FTDI SWD mode enabled
8 swd
9 Error: no device found
10 Error: unable to open ftdi device with vid 0403, pid 8a98, description '*', serial
   '*' at bus location '*'
11 Info : clock speed 2000 kHz
12 Error: Error connecting DP: cannot read IDR
13
14 ** OpenOCD init failed **
15 shutdown command invoked
```

Listing 5: Log when flashing the PCB fails.

4 The SWDIO pin for the micro-controller is `PA13`, but the header was connected to `PB3`. Upon cutting the trace with a scalpel and adding a jumper wire from the header to `PA13`, shown in Figure 21, the micro-controller could be successfully flashed (Listing 6). The trace cut was necessary as `PB3` is the JTAG reset pin, which could have prevented debugging. *JG,SH*

```
1 Open On-Chip Debugger 0.12.0
2 Licensed under GNU GPL v2
3 For bug reports, read
4   http://openocd.org/doc/doxygen/bugs.html
5 none separate
6
7 Info : FTDI SWD mode enabled
```

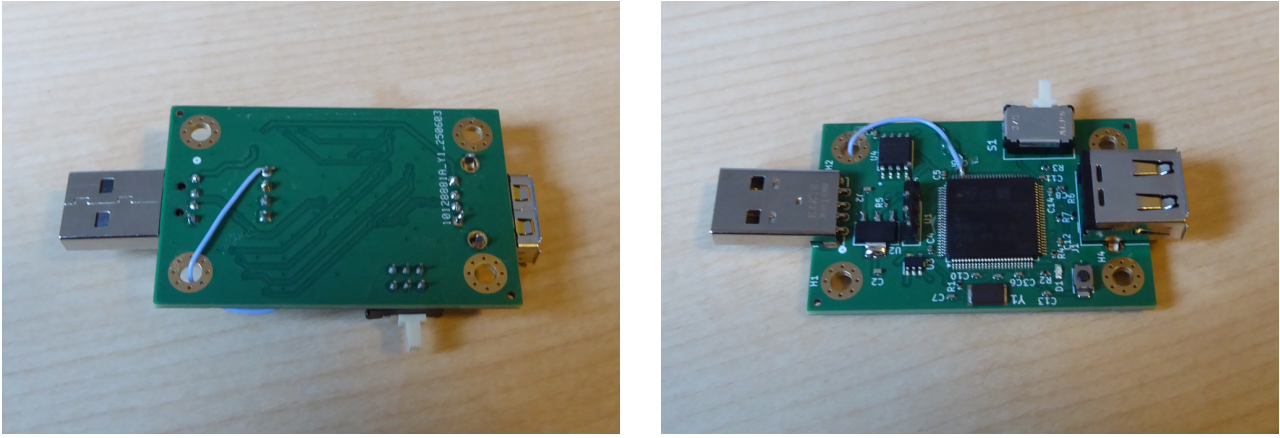


Figure 21: Modifications make to the PCB to fix SWD.

```

8  swd
9  Error: no device found
10 Error: unable to open ftdi device with vid 0403, pid 8a98, description '*', serial
    '*' at bus location '*'
11 Info : clock speed 2000 kHz
12 Info : SWD DPIDR 0x2ba01477
13 Info : [stm32f4x.cpu] Cortex-M4 r0p1 processor detected
14 Info : [stm32f4x.cpu] target has 6 breakpoints, 4 watchpoints
15 Info : starting gdb server for stm32f4x.cpu on 3333
16 Info : Listening on port 3333 for gdb connections
17 [stm32f4x.cpu] halted due to debug-request, current mode: Thread
18 xPSR: 0x01000000 pc: 0x08000274 msp: 0x20020000
19 ** Programming Started **
20 Info : device id = 0x101f6413
21 Info : flash size = 1024 KiB
22 ** Programming Finished **
23 ** Verify Started **
24 ** Verified OK **
25 ** Resetting Target **
26 Info : Listening on port 6666 for tcl connections
27 Info : Listening on port 4444 for telnet connections
28 shutdown command invoked

```

Listing 6: Log after successfully flashing the PCB.

7.3 Clock

1 Although the program in Listing 24 works, it is all but useless in practise. To do anything useful the clock scalars need configuring. These control the speeds the core and interfaces operate at. `libopencm3` [57] provides the `rcc` application interface (API) containing preconfigured clock setups for common external oscillators. The 8 MHz to 168 MHz¹ clock scalar was chosen. The template program for the following tests can be seen in Listing 25. *JG*

7.4 LED

1 The LED is connected to PA2 and sinks its current, which requires it to be configured to use the internal pull-down resistor. The test program, defined in Algorithm 1, makes the LED blink at a regular interval. The test program can be seen in Listing 26 and running in Figure 22. *JG*

7.5 Switch

1 The switch is connected to PA3 which inputs a logical 1 or 0 when in the left or right positions respectively. The test program, defined in Algorithm 2, turns the LED on or off when the switch is in the left or right positions respectively. The test program can be seen in Listing 27. *JG*

¹https://libopencm3.org/docs/latest/stm32f4/html/rcc_8c_source.html#l00156

Algorithm 1 Test LED Algorithm

```
procedure MAIN
  configure  $GPIOA_2 \leftarrow$  as pulldown output
  loop
    delay
     $GPIOA_2 \leftarrow \neg GPIOA_2$ 
  end loop
end procedure
```

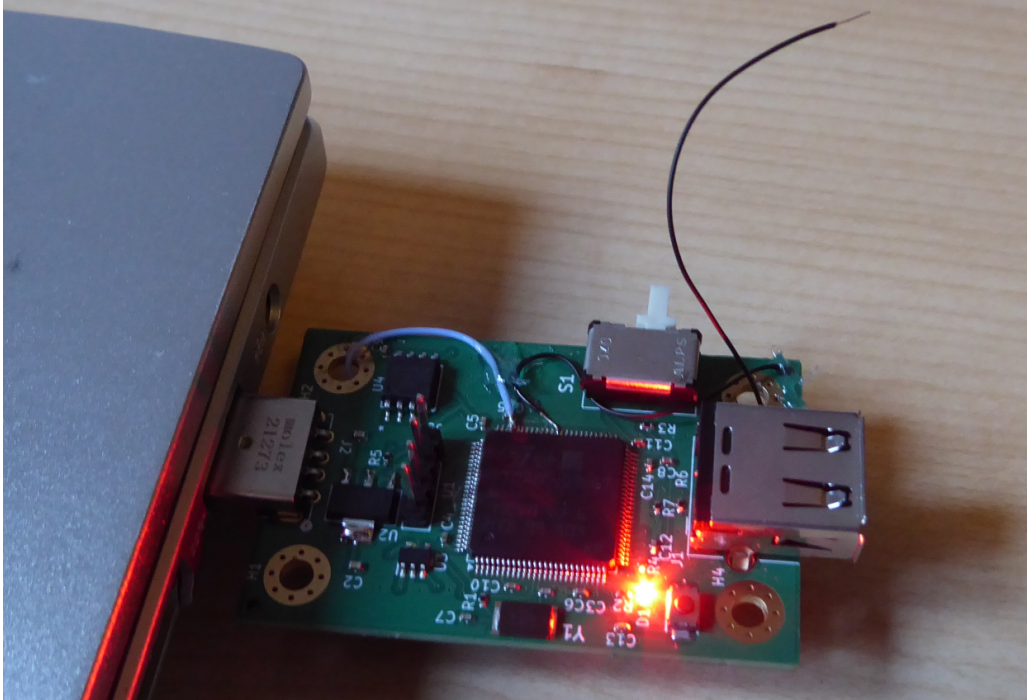


Figure 22: LED turning on during the test program.

Algorithm 2 Test Switch Algorithm

```
procedure MAIN
  configure  $GPIOA_2 \leftarrow$  as pulldown output
  configure  $GPIOA_3 \leftarrow$  as input
  loop
    if  $GPIOA_3 = 1$  then
       $GPIOA_2 \leftarrow 0$ 
    else
       $GPIOA_2 \leftarrow 1$ 
    end if
  end loop
end procedure
```

7.6 Serial

1 Serial was overlooked during the initial design of the PCB. It was possible to retroactively connect a wire to pin PC6 (Figure 23) which connects to the USART6 interface - a hardware module on the micro-controller for handling serial. The test program, defined in Algorithm 3, outputs "Hello World!" via serial. The test program can be seen in Listing 28. *JG,SH*

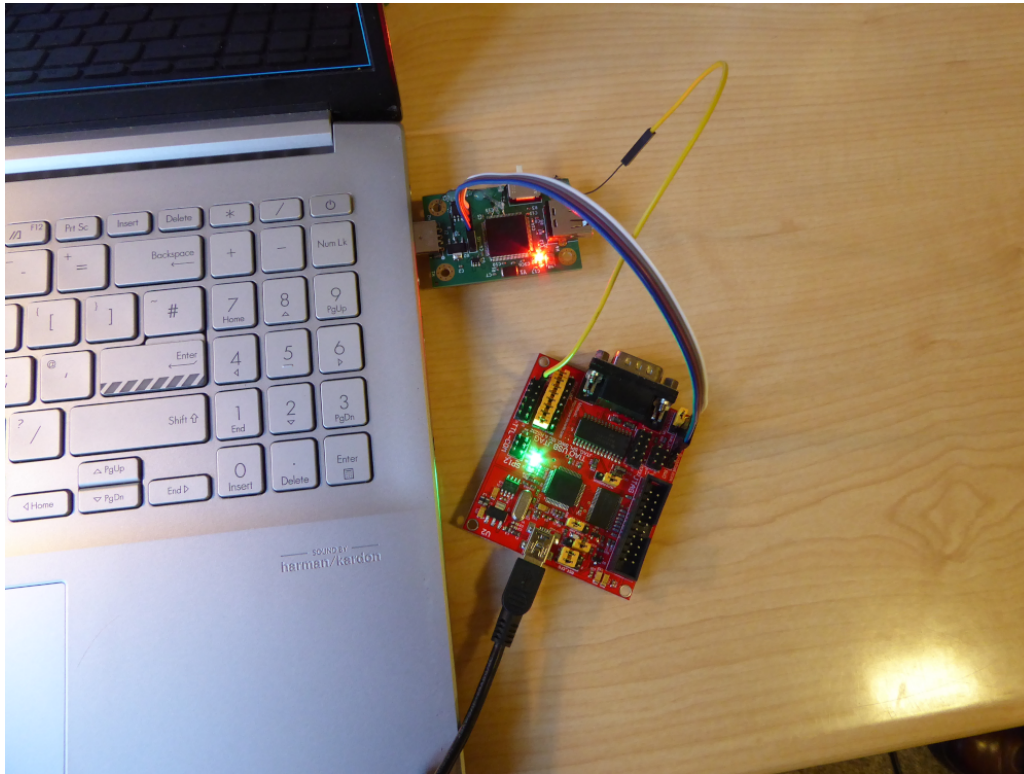


Figure 23: Setup showing serial connection.

Algorithm 3 Test Serial Algorithm

```
procedure MAIN
    configure USART6
    USART6  $\leftarrow$  "Hello World!"
    loop
    end loop
end procedure
```

7.7 Flash Memory

1 The flash is connected via the SPI1 interface to the micro-controller. The simplest method to test the SPI interface, is to read the Manufacture/Device ID from the flash and output it via serial. By sending the sequence 90 00 00 00 to the flash, the following two bytes should return the Manufacture ID (0xef) then the Device ID (0x17) as defined in Algorithm 4. The test program can be seen in Listing 29 and the serial response is shown in Listing 7. *JG*

```
1 picocom v2024-07
2
3 port is          : /dev/ttyUSB1
4 flowcontrol     : none
5 baudrate is     : 115200
6 parity is       : none
7 databits are    : 8
8 stopbits are    : 1
9 txdelay is      : 0 ns
10 escape is      : C-a
```

Algorithm 4 Test SPI Algorithm

```
procedure MAIN
  configure USART6
  configure SPI1
  devid,manid ← read_dev_man_id()
  USART6 ← “Device ID: ” devid
  USART6 ← “Manufacture ID: ” manid
  loop
  end loop
end procedure
```

```
11 local echo is : no
12 noinit is : no
13 noreset is : no
14 hangup is : no
15 nlock is : no
16 send_cmd is : sz -vv
17 receive_cmd is : rz -vv -E
18 imap is :
19 omap is :
20 emap is : crcrlf,delbs,
21 logfile is : none
22 initstring : none
23 exit_after is : not set
24 exit is : no
25 minimal cmds is: no
26
27 Type [C-a] [C-h] to see available commands
28 Terminal ready
29 Manufacture ID: 0xef
30 Device ID: 0x17
31
32 Terminating...
33 Thanks for using picocom
```

Listing 7: Result from SPI Program.

2 Testing the read and write functions of the SPI chip. Upon implementing the algorithm, the incorrect data was being outputted. This was due to the bytes being written needing returned bits to be read immediately after. This included parsing the address, writing data bytes and compiling data that is being read out. Algorithm 5 shows the pseudocode used, and the final code used in Listing 30. *SH*

7.8 USB

1 Upon writing the USB tests, it became apparent `libopencm3` was not adequate for this project. A different Hardware Access Layer (HAL) was required with a complete USB stack. This was provided by the `tinysub` project [58], which uses the `stm32f4xx_hal_driver` [59] as the HAL. *JG*

7.8.1 Board

1 The `tinysub` project comes with a large suite of example code for testing. These tests are built against a board configuration, which describes the USB interfaces available. For the micro-controller on the PCB, these are found in `hw/bsp/stm32f4/boards`². The board configuration closest to the PCB is the `stm32f407disco` [60], and even uses the exact same micro-controller. The LED and serial configurations were modified to match the PCB. *JG*

7.8.2 Device

1 The `hid_boot_interface`³ example was selected for testing the device USB connection. With this program flashed to the board and plugged into a laptop, nothing happened. Analysing the kernel logs from the

²<https://github.com/hathach/tinysub/tree/master/hw/bsp/stm32f4/boards>

³https://github.com/hathach/tinysub/tree/master/examples/device/hid_boot_interface

Algorithm 5 Test Flash Algorithm

```
procedure READ_FLASH
    select chip
    send read command
    read data
end procedure
procedure WRITE_FLASH
    enable writing
    read write enable bit
    if writing enabled then
        write to flash
    end if
    repeat
        read status register 1
    until status = 1
end procedure
procedure MAIN
    configure USART6
    configure SPI1
    WRITE_FLASH("Hello world")
    output ← READ_FLASH
    USART6 ← output
    loop
    end loop
end procedure
```

laptop, via `dmesg` [61], revealed that the laptop identified that the PCB was plugged in (Listing 8), but wasn't getting responses when it tried to communicate. *JG*

```
1 [ 218.042417] usb 1-3: new full-speed USB device number 3 using xhci_hcd
2 [ 218.162433] usb 1-3: device descriptor read/64, error -71
3 [ 218.383436] usb 1-3: device descriptor read/64, error -71
4 [ 218.601399] usb 1-3: new full-speed USB device number 4 using xhci_hcd
5 [ 218.721407] usb 1-3: device descriptor read/64, error -71
6 [ 218.944455] usb 1-3: device descriptor read/64, error -71
7 [ 219.047483] usb usb1-port3: attempt power cycle
8 [ 219.427419] usb 1-3: new full-speed USB device number 5 using xhci_hcd
9 [ 219.427524] usb 1-3: Device not responding to setup address.
10 [ 219.631471] usb 1-3: Device not responding to setup address.
11 [ 219.839379] usb 1-3: device not accepting address 5, error -71
12 [ 219.839474] usb 1-3: WARN: invalid context state for evaluate context command.
13 [ 219.952383] usb 1-3: new full-speed USB device number 6 using xhci_hcd
14 [ 219.952488] usb 1-3: Device not responding to setup address.
15 [ 220.159468] usb 1-3: Device not responding to setup address.
16 [ 220.368380] usb 1-3: device not accepting address 6, error -71
17 [ 220.368480] usb 1-3: WARN: invalid context state for evaluate context command.
18 [ 220.368530] usb usb1-port3: unable to enumerate USB device
```

Listing 8: `dmesg` logs when failing to connect the PCB.

2 The problem was identified by attaching an oscilloscope to the D+ and D- connections of the USB port and observing the signal in Figure 24. The signals sent by the laptop appear on both connections, but the responses from the PCB show only on the D- connection. It was found that VBUS sense was being enabled by the board configuration, left over from the `stm32f407disco`, when the PCB does not support it. *JG*

3 Still nothing happened when the PCB was connected to the laptop. Looking at the logs generated by `dmesg`, suggested that the device it was trying to connect to was LS, but the PCB had been designed for FS. The hardware configuration required to differentiate between the two differs only by which wire is pulled high, D+ or D- for HS and LS respectively. *JG,SH*

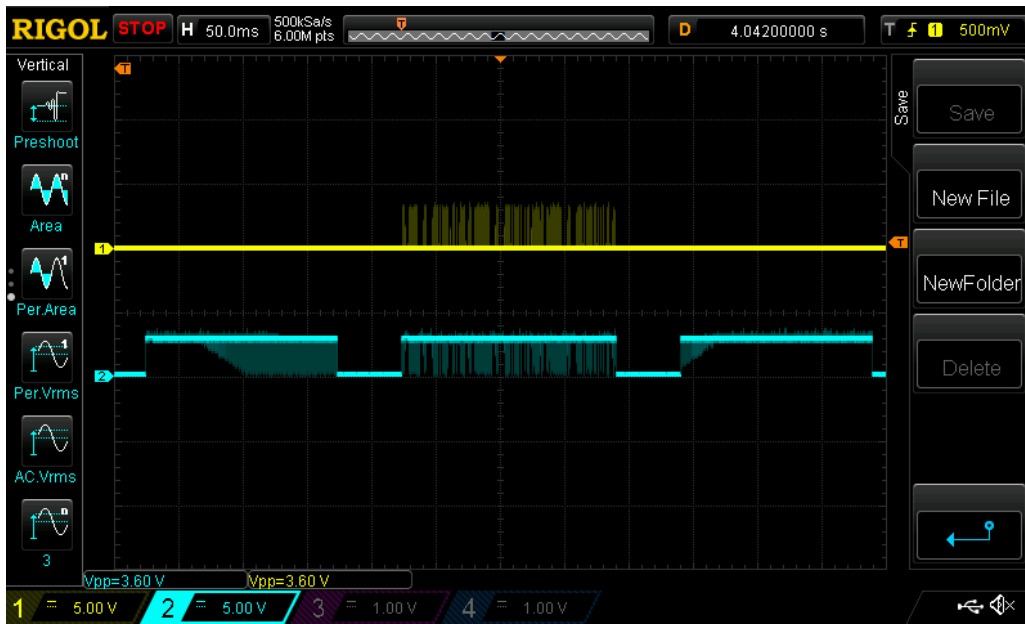


Figure 24: Oscilloscope showing bad USB.

4 This meant that D+ and D- were swapped. This was rectified using jumper wires to swap the lines, and severing the traces after the pull up resistor as shown in Figure 25. The `dmesg` log shows a successful connection in Listing 9. This fundamental problem with the PCB led to the decision being taken to make a redesign. *JG,SH*

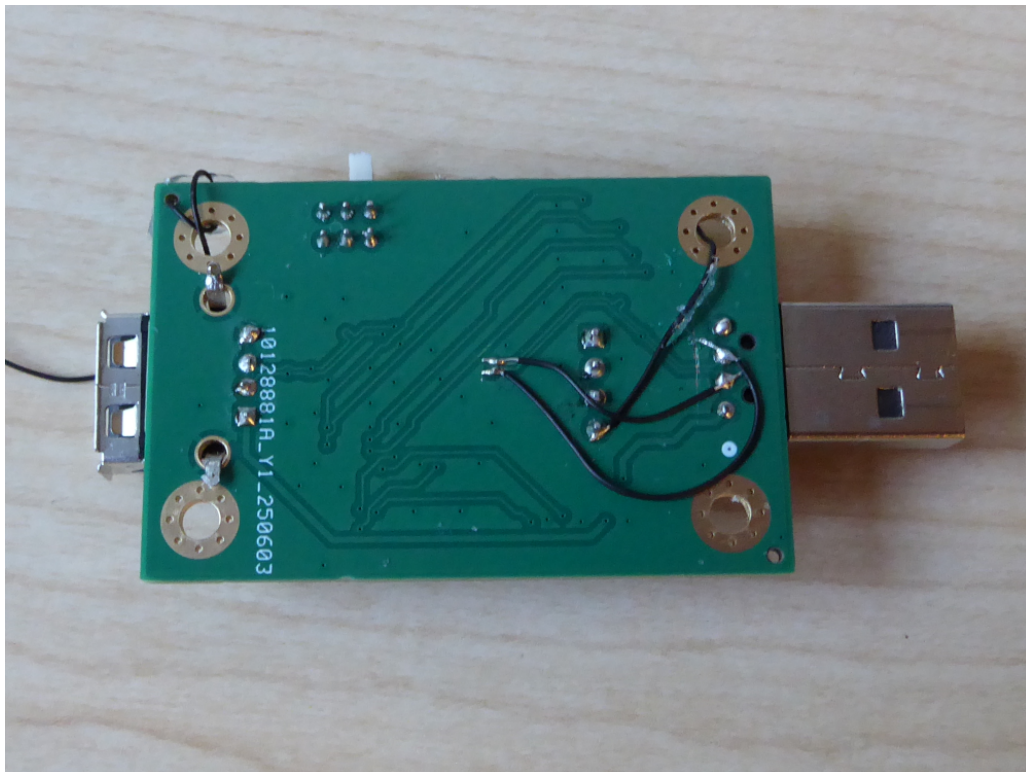


Figure 25: Modifications make to the PCB to fix USB.

```

1 [ 1428.048344] usb 1-3: new full-speed USB device number 3 using xhci_hcd
2 [ 1428.211416] usb 1-3: New USB device found, idVendor=2a2a, idProduct=4004,
   bcdDevice= 1.00
3 [ 1428.211428] usb 1-3: New USB device strings: Mfr=1, Product=2, SerialNumber=3
4 [ 1428.211435] usb 1-3: Product: CKAKL Device
5 [ 1428.211441] usb 1-3: Manufacturer: CKAKL

```

```

6 [ 1428.211446] usb 1-3: SerialNumber: 3B0059001451343036363231
7 [ 1428.283147] input: CKAKL CKAKL Device as /devices/pci0000
  :00/0000:00:08.1/0000:03:00.3/usb1/1-3/1-3:1.0/0003:2A2A:4004.0002/input/input16
8 [ 1428.359652] hid-generic 0003:2A2A:4004.0002: input,hidraw1: USB HID v1.11 Keyboard
  [CKAKL CKAKL Device] on usb-0000:03:00.3-3/input0
9 [ 1428.359742] usbcore: registered new interface driver usbhid
10 [ 1428.359748] usbhid: USB HID core driver

```

Listing 9: dmesg logs when successfully connecting the PCB.

7.8.3 Host

1 The `cdc_msc_hid`⁴ example was selected for testing the host USB connection. With this program flashed to the board, powered, and a keyboard connected, nothing happened. Checking the serial output showed that keyboard was not detected. *JG*

2 The problem was identified when attempting to connect a mouse, in case the keyboard had a fault, it was noted that the optical LED was not turning on. This revealed that the **VBUS** was not providing power to the devices due to an oversight in the board configuration. This was fixed by enabling the **VBUS** upon host initialisation. *JG,SH*

7.8.4 Both

1 The `host_info_to_device_cdc`⁵ example was selected for testing the host USB connection. With this program flashed to the board, powered, a keyboard connected, and a monitor on `/dev/ttyACMO`, everything worked correctly. *JG*

⁴https://github.com/hathach/tinyusb/tree/master/examples/host/cdc_msc_hid

⁵https://github.com/hathach/tinyusb/tree/master/examples/dual/host_info_to_device_cdc

7.9 New Board Design

1 Once a new board design was required, the design fixes mentioned in the sections above were reflected in the schematic. Figure 2 below shows the erroneous SWD pins and corrected pins within KiCADs schematic design environment. For future use, pads that could be used to probe onto the UART interface were added to the schematic, which would allow for easier connection than attaching wires directly to the chip. The USB differential pair was swapped and an addition resistor pad added to ensure that the resistor pulling up the line could be replaced easily if needed. *SH*

2 A new PCB design was created, Figure 27, allowing for a new design to be created with the knowledge and experience gathered from the previous design process. Only two mounting holes were added, reducing the space needed, alongside reducing the spacing components and using a more efficient layout of the components. The printed circuit is shown in Figure 28. *SH*

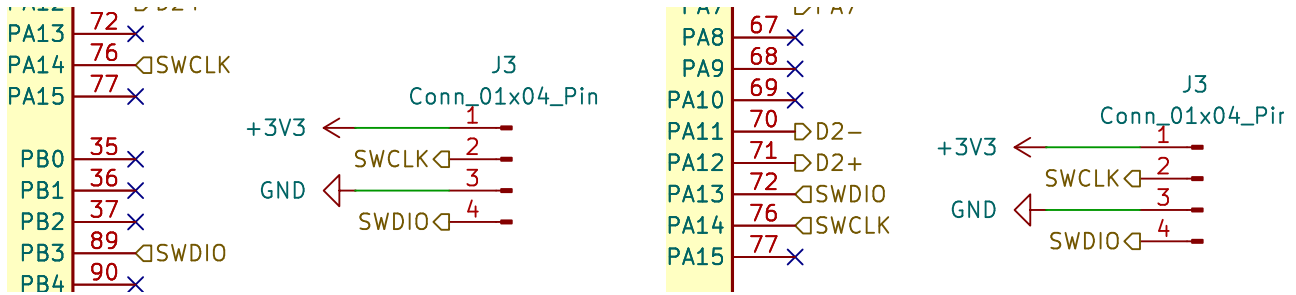


Figure 26: Erroneous schematic.

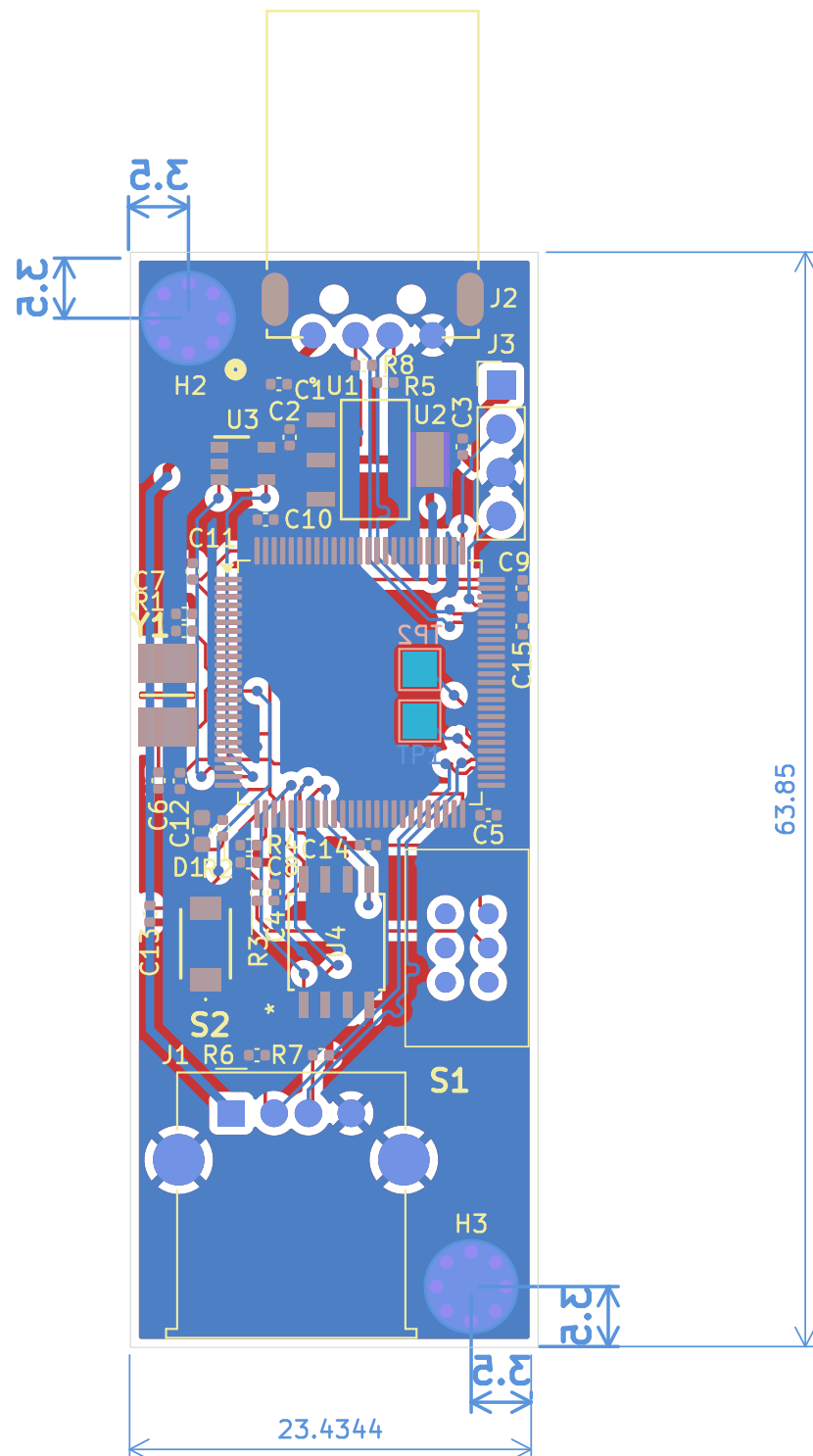


Figure 27: Final PCB design.

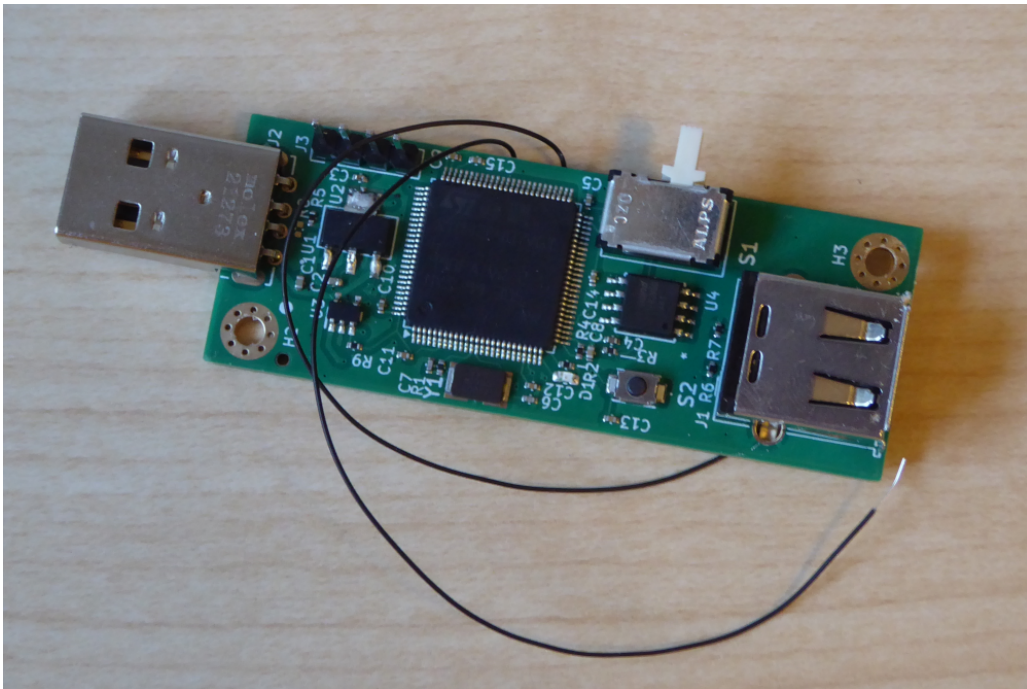


Figure 28: Final board desgin.

8 Software

8.1 YAML Parser

1 To enable the user to configure the device through the use of a YAML file, a parser is required to read the file and store data into appropriate internal structures. The open-source library `libcyaml` [62] provides schema based YAML parsing and serialisation. The flowchart for the parser is shown in Figure 29. *BG* [JG] ⇔

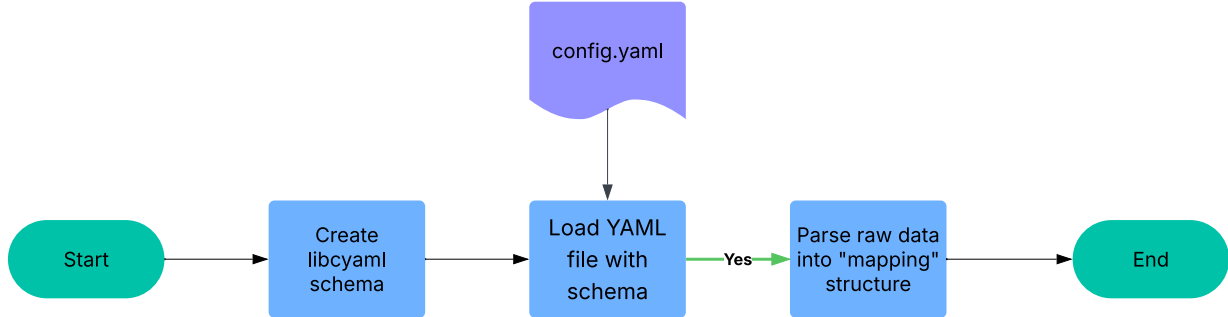


Figure 29: High-level flowchart diagram of YAML configuration file parser and string tokeniser.

8.1.1 Implementation

1 The first stage of programming the parser is to define the schema. The schema is a set structure that is used to describe the layout and contents of the YAML file being parsed. Shown in Listing 10 is an example YAML file that can be parsed by the program. *BG*

```
1 device_id: 1234
2 vendor_id: 5678
3 mappings:
4   - "A:B"
5   - "CTRL:SHIFT"
```

Listing 10: Example YAML file for configuring the device.

2 As shown in Listing 10, `device_id` and `vendor_id` are simple scalar values, whereas `mappings` is a sequence of strings, or in C referred to as `char *`. These values are to be stored in a structure called `yaml_file` (Listing 11) which will later be used for post-processing the raw data. Shown in Listing 12 is this top level schema description using the macros provided by the `libcyaml` library. *BG*

```
1
2 struct yaml_file {
3     unsigned int did;
4     unsigned int vid;
5     unsigned int raw_mappings_count;
6     char** raw_mappings;
7 };
```

Listing 11: `yaml_file` structure definition.

```
1
2 static const cyaml_schema_value_t mapping_entry = {
3     CYAML_VALUE_STRING(CYAML_FLAG_POINTER, char, 0, CYAML_UNLIMITED),
4 };
5
6 static const cyaml_schema_field_t top_mapping_schema[] = {
7     CYAML_FIELD_UINT("device_id", CYAML_FLAG_DEFAULT, struct yaml_file, did),
8     CYAML_FIELD_UINT("vendor_id", CYAML_FLAG_DEFAULT, struct yaml_file, vid),
9     CYAML_FIELD_SEQUENCE("mappings", CYAML_FLAG_POINTER | CYAML_FLAG_OPTIONAL, struct
        yaml_file, raw_mappings, &mapping_entry, 0, CYAML_UNLIMITED),
```

```

10     CYAML_FIELD_END
11 };
12
13 static const cyaml_schema_value_t top_schema = {
14     CYAML_VALUE_MAPPING(CYAML_FLAG_POINTER, struct yaml_file, top_mapping_schema),
15 };

```

Listing 12: Top level schema using libcyaml macros.

3 The final step is to call the `cyaml_load_file` function and pass in the name of the configuration file, settings for libcyaml to use, the schema created previously, and a pointer to a structure matching the schema. This function returns a success code, so a simple conditional check is performed to handle any errors. This code can be seen in Listing 13. *BG*

```

1
2     err = cyaml_load_file("config.yaml", &config, &top_schema, (cyaml_data_t **)&yf,
3         NULL);
4     if (err != CYAML_OK) {
5         fprintf(stderr, "ERROR: %s\n", cyaml_strerror(err));
6         return EXIT_FAILURE;
7     }

```

Listing 13: Calling `cyaml_load_file` and handling the return status.

4 The contents of the YAML configuration file is now stored within the `yaml_file` structure within the program. *BG*

8.1.2 Data Extraction

1 The next stage of the YAML parser is to take the raw data read from the YAML configuration file and tokenise it into a useful structure (Listing 14) that can be used later in the program. *BG*

```

1
2 struct mapping {
3     unsigned count;
4     char** from;
5     char** to;
6 };

```

Listing 14: `mapping` structure to store keyboard mappings from YAML configuration file.

2 This tokenisation is performed by iterating over the `raw_mappings` member of the `yaml_file` structure. Each string in the list is split by using a `:` as the delimiter. This allows the key being mapped to be stored in the `from` field of the `mapping` structure and the key it is being mapped to being stored in the `to` field. Finally, the amount of mappings present in the YAML configuration file is copied into the `count` member of the `mapping` structure. This function can be seen in Listing 15. *BG*

```

1
2 void parse_raw_mappings(struct yaml_file* file, struct mapping* mapping) {
3     char* tmp;
4     mapping->from = malloc(file->raw_mappings_count * sizeof(char*));
5     mapping->to = malloc(file->raw_mappings_count * sizeof(char*));
6     for (unsigned i = 0; i < file->raw_mappings_count; i++) {
7         tmp = strtok(file->raw_mappings[i], ":");
8         mapping->from[i] = malloc(strlen(tmp) + 1);
9         strcpy(mapping->from[i], tmp);
10        tmp = strtok(NULL, ":");
11        mapping->to[i] = malloc(strlen(tmp) + 1);
12        strcpy(mapping->to[i], tmp);
13    }
14    mapping->count = file->raw_mappings_count;
15 }

```

Listing 15: Tokenisation of the mappings present in the YAML configuration file.

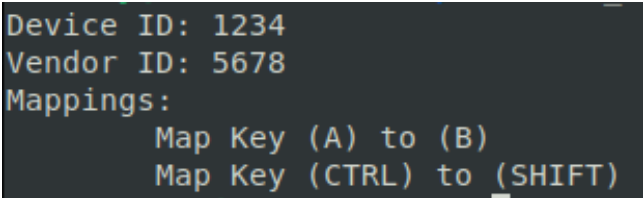
3 The data is now stored in the `mappings` structure in a logical manner enabling future use to be more efficient and concise. *BG*

8.1.3 Testing

1 To test the YAML configuration file parser and string tokeniser, a simple configuration file was created (Listing 10) and an iterator was used to loop through the mappings and print the values to the terminal. Shown in Listing 16 is the iterator and Figure 30 shows the output from the terminal. *BG*

```
1 void print_mappings(const struct mapping* mapping, const struct yaml_file* yf) {
2     printf("Device_ID: %u\n", yf->did);
3     printf("Vendor_ID: %u\n", yf->vid);
4     printf("Mappings:\n");
5     for (unsigned i = 0; i < mapping->count; i++) {
6         printf("\tMap_Key (%s) to (%s)\n", mapping->from[i], mapping->to[i]);
7     }
8 }
9 }
```

Listing 16: Iterator to print mappings to terminal.



```
Device ID: 1234
Vendor ID: 5678
Mappings:
    Map Key (A) to (B)
    Map Key (CTRL) to (SHIFT)
```

Figure 30: Output from the terminal.

8.1.4 Errors

1 Within the development of this piece of software, some errors were encountered and fixes were implemented to ensure the software functioned as intended. *BG*

2 When parsing the YAML configuration file with the `libcyaml` library, the list of strings present that represent the keyboard mappings was not being parsed correctly. Within the definition of the schema for the file, a macro called `CYAML_FIELD_SEQUENCE` is being used to describe this list of strings. However, the list and amount of elements present are classed as optional fields within the YAML file. To fix the macro, the `CYAML_FLAG_OPTIONAL` flag needed to be set as shown in Listing 17. *BG*

```
1
2     CYAML_FIELD_SEQUENCE("mappings", CYAML_FLAG_POINTER | CYAML_FLAG_OPTIONAL, struct
        yaml_file, raw_mappings, &mapping_entry, 0, CYAML_UNLIMITED),
```

Listing 17: Setting the `CYAML_FLAG_OPTIONAL`.

3 Additionally, two errors were present due to memory allocation when de-referencing pointers and attempting to store a string (`char *`) value at the address. These errors were fixed sequentially by making a function call to `malloc` with the size of memory to be allocated. The pointer returned from the function is then assigned to appropriate pointer variable or structure member. These two fixes can be seen in Listing 18 and Listing 19 respectively. *BG*

```
1
2     tmp = strtok(file->raw_mappings[i], ":");
3     mapping->from[i] = malloc(strlen(tmp) + 1);
4     strcpy(mapping->from[i], tmp);
5     tmp = strtok(NULL, ":");
6     mapping->to[i] = malloc(strlen(tmp) + 1);
7     strcpy(mapping->to[i], tmp);
```

Listing 18: `malloc` function call to allocate memory for storing strings representing the keyboard remappings.

```

1 mapping->from = malloc(file->raw_mappings_count * sizeof(char*));
2 mapping->to = malloc(file->raw_mappings_count * sizeof(char*));
3

```

Listing 19: malloc function calls to allocate memory for the list of string pointers (char **).

8.2 Remapper

8.2.1 Passthrough

1 Before adding the remappings, it was decided to make the program accept keyboard packets from its host side and transfer them out the device side to the laptop. This algorithm is defined in Algorithm 6. **JG**

Algorithm 6 Passthrough Algorithm

```

procedure MAIN
  configure  $USB_{HS}$   $\leftarrow$  as keyboard
  configure  $USB_{FS}$   $\leftarrow$  as host to keyboard
  loop
    if have  $USB_{FS}$  packet then
      send  $USB_{HS} \leftarrow USB_{FS}$  packet
    end if
  end loop
end procedure

```

2 To achieve this with tinyusb callbacks are used to process key events. Performing actions on peripheral packets requires the tuh_hid_report_received_cb callback as shown in Listing 20. **JG**

```

1
2 // Invoked when received report from device via interrupt endpoint
3 void tuh_hid_report_received_cb(uint8_t dev_addr, uint8_t instance, uint8_t const *
4   report, uint16_t len) {
5   uint8_t const itf_protocol = tuh_hid_interface_protocol(dev_addr, instance);
6
7   switch (itf_protocol) {
8     case HID_ITF_PROTOCOL_KEYBOARD:
9       TU_LOG2("HIDreceivebootkeyboardreport\r\n");
10      process_kbd_report((hid_keyboard_report_t const *) report);
11      break;
12
13     case HID_ITF_PROTOCOL_MOUSE:
14       TU_LOG2("HIDreceivebootmousereport\r\n");
15       process_mouse_report((hid_mouse_report_t const *) report);
16       break;
17
18     default:
19       // Generic report requires matching ReportID and contents with previous parsed
20       // report info
21       process_generic_report(dev_addr, instance, report, len);
22       break;
23   }
24
25   // continue to request to receive report
26   if (!tuh_hid_receive_report(dev_addr, instance)) {
27     printf("Error: cannot request to receive report\r\n");
28   }
29 }
30
31 //-----+
32 // Keyboard
33 //-----+
34
35 static void process_kbd_report(hid_keyboard_report_t const *report) {
36   send_kbd_report(report);
37 }

```

35 }

Listing 20: tuh_hid_report_received_cb implementation.

3 send_kbd_report, Listing 21, then sends the packet to the computer on the device side. This also handles the case where the computer has gone to sleep and needs to be woken up first, via tud_resume_cb. **JG**

```

1
2 static hid_keyboard_report_t const *to_send = NULL;
3
4 // Invoked when usb bus is resumed
5 void tud_resume_cb(void)
6 {
7     if (to_send != NULL) {
8         tud_hid_keyboard_report(REPORT_ID_KEYBOARD, to_send->modifier, to_send->keycode);
9         to_send = NULL;
10    }
11 }
12
13 //-----+
14 // USB HID
15 //-----+
16
17 void send_kbd_report(hid_keyboard_report_t const *report)
18 {
19     // skip if hid is not ready yet
20     if ( !tud_hid_ready() ) return;
21
22     // Remote wakeup
23     if ( tud_suspended() )
24     {
25         to_send = report;
26         // Wake up host if we are in suspend mode
27         // and REMOTE_WAKEUP feature is enabled by host
28         tud_remote_wakeup();
29     }else
30     {
31         tud_hid_keyboard_report(0, report->modifier, report->keycode);
32     }
33 }

```

Listing 21: send_kbd_report implementation.

8.2.2 Configuration

1 Due to time constraints at this point it was not possible to implement the file system on the flash and instead the configuration was hard-coded into the program. This algorithm is defined in Algorithm 7. **JG**

Algorithm 7 Remapping Algorithm

```

remappings ← []
procedure MAIN
    configure  $USB_{HS}$  ← as keyboard
    configure  $USB_{FS}$  ← as host to keyboard
    loop
        if have  $USB_{FS}$  packet then
             $p \leftarrow \text{remappings}[USB_{FS} \text{ packet}]$ 
            send  $USB_{HS} \leftarrow p$ 
        end if
    end loop
end procedure

```

2 The file `kbd_remap.h` contains an array with the remappings call `kbd_remaps`. Listing 22 shows the new implementation of `send_kbd_report`. The working setup can be seen in Figure 31. *JG*

```
1
2 static hid_keyboard_report_t const *to_send = NULL;
3
4 // Invoked when usb bus is resumed
5 void tud_resume_cb(void)
6 {
7     if (to_send != NULL) {
8         tud_hid_keyboard_report(REPORT_ID_KEYBOARD, to_send->modifier, to_send->keycode);
9         to_send = NULL;
10    }
11 }
12
13 //-----+
14 // USB HID
15 //-----+
16
17 void send_kbd_report(hid_keyboard_report_t const *report)
18 {
19     // skip if hid is not ready yet
20     if ( !tud_hid_ready() ) return;
21
22     // Remote wakeup
23     if ( tud_suspended() )
24     {
25         to_send = report;
26         // Wake up host if we are in suspend mode
27         // and REMOTE_WAKEUP feature is enabled by host
28         tud_remote_wakeup();
29     }else
30     {
31         uint8_t keycodes[] = {0,0,0,0,0,0};
32         for (unsigned i = 0; i < 6; i++) {
33             keycodes[i] = kbd_remaps[report->keycode[i]];
34         }
35         tud_hid_keyboard_report(0, report->modifier, keycodes);
36     }
37 }
```

Listing 22: `send_kbd_report` implementation with remappings.



Figure 31: Working setup with keyboard.

9 Evaluation

1 Figure 32 shows the final Gantt chart for the project. **SH**

2 The cost of the production for boards and the necessary parts fell well below the set budget, which is a resounding success as shown in Figure 33. **SH**

3 The hardware fulfils the following criteria in the specification:

- Able to sit in series of a keyboard and host machine and not effect keyboard functionality.
- Have a physical switch to alternate between active and configuration modes.
- Comply with Electronic Equipment (safety) regulations 2016.
- Comply with RoHS regulations.
- Comply with WEEE regulations 2013.
- Comply with the Electromagnetic Compatibility regulations 2016.
- Provide USB 2.0 compatibility.
- Contain and display error light sequences to notify the user of particular errors.

SH

4 The hardware works as intended, with no modifications required. Its fulfils most of the requirements set out in the initial specification, which can be classed as a success. **SH**

5 The YAML configuration file parser and string tokeniser successfully enables the configuration of the device by the user, satisfying the requirements set out. However, the software has not been tested on the actual device, nor has a filesystem been created due to the hardware testing taking longer than expected. If the project were to be repeated, more time should be allocated to the hardware testing and greater care taken in reading appropriate documentation to eliminate some of the errors. Additionally, the software development could take place on a development board to avoid the extra step of porting it to the embedded hardware. **BG**

6 The hardware has been thoroughly tested by an extensive testing suite of programs. A working prototype was able to be completed, that allowed for limited remapping of the keyboard, but the requirements for the software were not met. **JG**

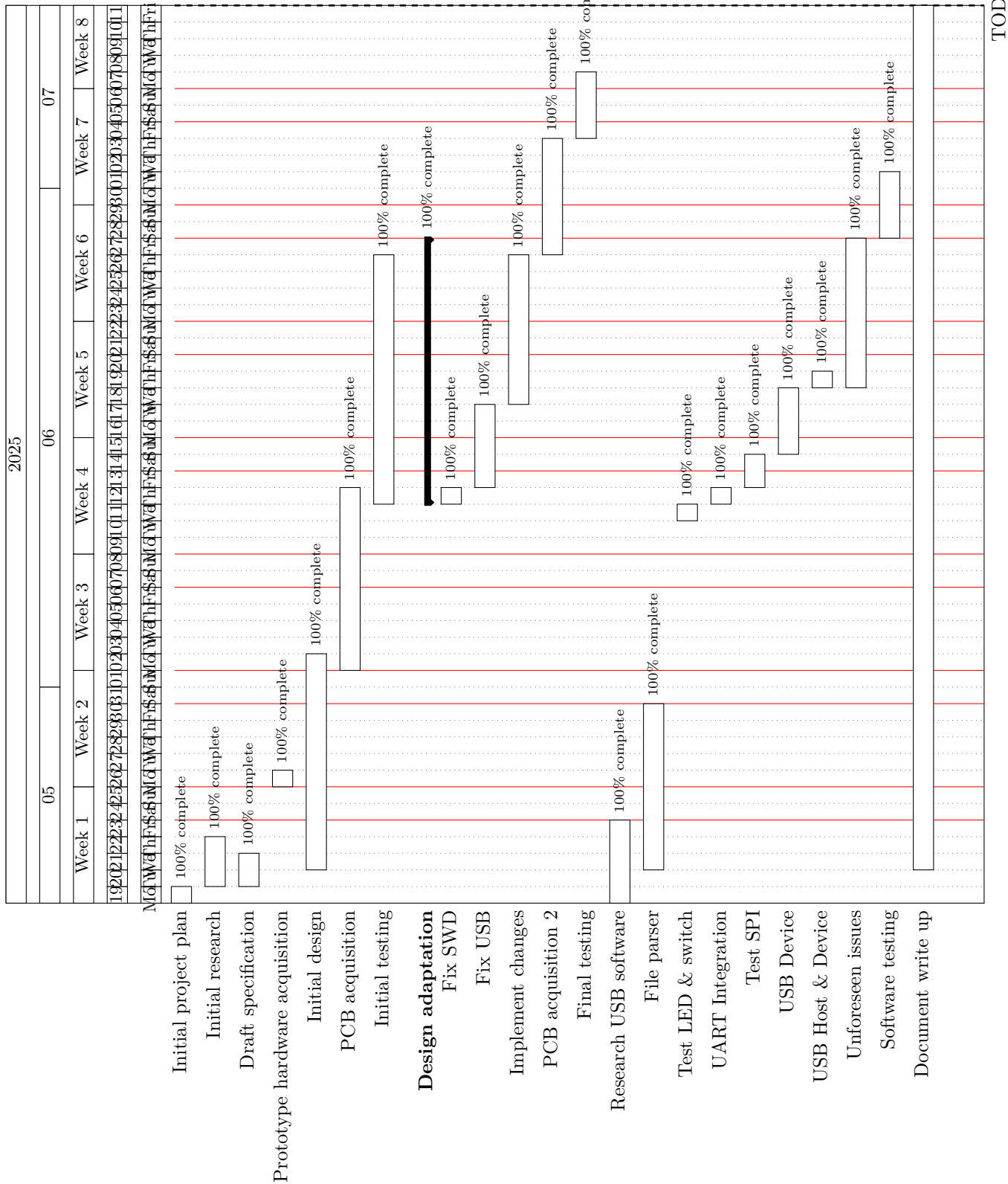


Figure 32: Gantt chart for this project.

Name	Cost (£)
PCB1	65.75
PCB1 parts	36.25
PCB2	67.24
Total	169.24

Figure 33: Full costing excluding time.

10 Conclusion

1 The modifications required to get the initial boards to work massively hindered the development of the software. In conjunction with the lack of development hardware, this meant that the software required to fulfil all of the specification in time is improbable. However, the ground work is still in place to complete the code fully at a later date. Although the project initially started ahead of schedule, it quickly fell behind. ***SH***

2 A YAML file parser was created using the open-source and free to use library `libcyaml`. A string tokeniser was added to parse the data read from the configuration file into meaningful data that can be further processed. This enables the user to configure the device with ease, satisfying the requirement set out earlier in this report. ***BG***

3 Initial setbacks due to problems with the PCB meant that the software was not able to be completed within the timeframe of this project. However, fixing the PCB problems demonstrated how well all members worked together. ***JG***

11 Recommendations

1 In the future, it would be apt to further downsize the board design using more advanced but complicated and expensive manufacturing methods, like 4 layered, a smaller processor and double sided placement on the PCBs. This would allow for a more portable form factor, which would likely fulfil the consumer appetite for such a product. ***SH***

2 Further work would include porting the software to be functional on the embedded device and providing a FAT16 filesystem so a computer can read the data stored on the device. ***BG***

3 Having a hardcoded configuration was not part of the original requirements, which stated that it should be stored on a FAT16 file system. The next steps would be to implement this. ***JG***

12 Reflection

1 For future reference, I think it would be apt to spend more time on the design to negate the risk of small mistakes falling through to the produced design. This would've reduced the time that was lost fixing mistakes and increase the time that could be used for software development. Also, in future endeavours it would be highly beneficial to leverage coworkers. **SH**

2 As an individual I learnt that researching the problem and solutions up front provides a clearer approach to tackling the project. It enables a clear understanding of what needs to be done and how we are going to achieve it. Additionally, engaging as a project team regularly with weekly meetings enables problems to be identified sooner and appropriate action plans to address the issues to be put in place. If I were to do the project again, I would develop the software solution targeting the embedded system from the start to eliminate the need for an extra step towards to the end of the project. Furthermore, I would allow more time to perform the hardware testing as this is what delayed the completion of the overall project. **BG**

3 Whilst doing this project, I have learnt a lot about diagnosing and fixing hardware problems and issues, and also identifying which tool is the most suitable. The collaboration and teamwork was a great help and enabled me to get as much done as I did. If I was to do this again, I would use my new hardware skills to help remove more problems earlier, to save the headaches and slow-ups later down the line. **JG**

13 References

- [1] “Keyboard Manager utility,” Microsoft. (2024), [Online]. Available: <https://learn.microsoft.com/en-us/windows/powertoys/keyboard-manager>. (accessed: 20.05.2025).
- [2] I. U. P. Kamil Toman, *How to further enhance XKB configuration*, X.Org Foundation, 2002. [Online]. Available: <https://www.x.org/releases/current/doc/xorg-docs/input/XKB-Enhancing.pdf>, (accessed: 20.05.2025).
- [3] paultheman. “Guide on how to remap Keyboard keys on macOS.” (2024), [Online]. Available: <https://gist.github.com/paultheman/808be117d447c490a29d6405975d41bd>. (accessed: 20.05.2025).
- [4] mnottale. “Hardware keyboard remapper.” (2025), [Online]. Available: <https://github.com/mnottale/keyboard-mapper/>. (accessed: 20.05.2025).
- [5] 41J. “A Programmable Keyboard Controller.” (2011), [Online]. Available: <https://41j.com/blog/2011/10/a-programmable-keyboard-controller/>. (accessed: 20.05.2025).
- [6] darkytoothpaste. “Arduino-based keyboard layout remapper.” (2013), [Online]. Available: <https://github.com/darkytoothpaste/keymapper/>. (accessed: 20.05.2025).
- [7] “HID Remapper.” (2025), [Online]. Available: <https://www.remapper.org/>. (accessed: 20.05.2025).
- [8] “AutoHotkey,” AutoHotKey. (2025), [Online]. Available: <https://www.autohotkey.com/>. (accessed: 20.05.2025).
- [9] Whitson Gordon and Jason Cohen. “Shortcuts, Hotkeys, Macros, Oh My: How to Remap Your Keyboard,” PC Magazine. (2025), [Online]. Available: <https://www.pcmag.com/how-to/remap-your-keyboard>. (accessed: 09.07.25).
- [10] “How to COPY keyboard assignments/mapping,” AVSIM. (2022), [Online]. Available: <https://www.avsim.com/forums/topic/621284-how-to-copy-keyboard-assignmentsmapping/>. (accessed: 09.07.25).
- [11] S. Fernandes, *xHCI Interoperability Test Procedures For Peripherals, Hubs and Hosts (Legacy, USB Type-C and Power Delivery)*, USB Implementers Forum, 2024. [Online]. Available: https://www.usb.org/sites/default/files/3.2%20Interoperability%20Testing%20v1.03%20w%20USB%20Type-C_0.pdf, (accessed: 20.05.25).
- [12] *STM32F103x8, STM32F103xB*, STMicroelectronics, 2023. [Online]. Available: <https://www.st.com/resource/en/datasheet/stm32f103cb.pdf>, (accessed: 20.05.25).
- [13] *STM32F303xD, STM32F303xE*, STMicroelectronics, 2016. [Online]. Available: <https://www.st.com/resource/en/datasheet/stm32f303re.pdf>, (accessed: 20.05.25).
- [14] *STM32F405XX, STM32F407XX*, STMicroelectronics, 2024. [Online]. Available: <https://www.st.com/resource/en/datasheet/stm32f407vg.pdf>, (accessed: 20.05.25).
- [15] *PIC16F87XA*, MicroChip, 2013. [Online]. Available: <https://ww1.microchip.com/downloads/aemDocuments/documents/MCU08/ProductDocuments/DataSheets/39582C.pdf>, (accessed: 20.05.25).
- [16] “USB Communication with PIC Microcontroller,” MicroControllersLab. (2017), [Online]. Available: <https://microcontrollerslab.com/usb-interfacing-pic-microcontroller-code/>. (accessed: 20.05.25).
- [17] *PIC18F2455/2550/4455/4550 Data Sheet*, MicroChip, 2009. [Online]. Available: <https://ww1.microchip.com/downloads/aemDocuments/documents/OTH/ProductDocuments/DataSheets/39632e.pdf>, (accessed: 20.05.25).
- [18] *ATmega16U4/ATmega32U4*, Atmel, 2016. [Online]. Available: https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7766-8-bit-AVR-ATmega16U4-32U4_Datasheet.pdf, (accessed: 20.05.25).
- [19] *FT4232H Quad High-Speed USB to Multi-Purpose UART/MPSSSE IC*, FTDI, 2024. [Online]. Available: https://ftdichip.com/wp-content/uploads/2024/09/DS_FT4232H.pdf, (accessed: 20.05.25).
- [20] *S25FL128S/S25FL256S*, Cypress, 2025. [Online]. Available: <https://docs.rs-online.com/971f/0900766b8167cd2e.pdf>, (accessed: 22.05.25).
- [21] *S25FL256L/S25FL128L*, Cypress, 2025. [Online]. Available: <https://docs.rs-online.com/73d7/0900766b816ccfe9.pdf>, (accessed: 22.05.25).
- [22] *S25HS256T/S25HS512T/S25HS01GT/S25HL256T/S25HL512T/S25HL01GT*, Cypress, 2025. [Online]. Available: <https://docs.rs-online.com/3b0f/A700000006654164.pdf>, (accessed: 22.05.25).
- [23] *3V 32M-BIT SERIAL FLASH MEMORY WITH DUAL, QUAD SPI*, Winbon, 2025. [Online]. Available: <https://docs.rs-online.com/0d6a/0900766b81622f91.pdf>, (accessed: 22.05.25).

- [24] *3V 128M-BIT SERIAL FLASH MEMORY WITH DUAL/QUAD SPI*, Winbon, 2025. [Online]. Available: <https://docs.rs-online.com/cdee/0900766b81622f85.pdf>, (accessed: 22.05.25).
- [25] "Difference Between YAML and JSON," GeeksForGeeks. (2024), [Online]. Available: <https://www.geeksforgeeks.org/what-is-the-difference-between-yaml-and-json/>. (accessed: 20.05.25).
- [26] "Remapping the keyboard," IBM. (2022), [Online]. Available: <https://www.ibm.com/docs/en/host-on-demand/15.0.x?topic=sessions-remapping-keyboard>. (accessed: 20.05.25).
- [27] "What Are Macro Keys and How to Use Them for Gaming," HP. (2024), [Online]. Available: <https://www.hp.com/gb-en/shop/tech-takes/what-are-macro-keys>. (accessed: 20.05.25).
- [28] "How To Use Multimedia and Function Keys on Dell External Keyboards," Dell Technologies. (2025), [Online]. Available: <https://www.dell.com/support/kbdoc/en-uk/000205137/how-to-use-the-multimedia-and-function-keys-on-dell-external-multimedia-keyboards>. (accessed: 20.05.25).
- [29] "Keyboard Input Overview," Microsoft. (2023), [Online]. Available: <https://learn.microsoft.com/en-us/windows/win32/inputdev/about-keyboard-input>. (accessed: 20.05.25).
- [30] *The Electrical Equipment (Safety) Regulations 2016*, UK Government, 2016. [Online]. Available: <https://www.legislation.gov.uk/ukxi/2016/1101/contents>, (accessed: 09.07.25).
- [31] *Regulations: restriction of hazardous substances (RoHS)*, UK Government, 2014. [Online]. Available: <https://www.gov.uk/guidance/rohs-compliance-and-guidance>, (accessed: 09.07.25).
- [32] "Regulations: waste electrical and electronic equipment (WEEE)," UK Government. (2018), [Online]. Available: <https://www.gov.uk/guidance/regulations-waste-electrical-and-electronic-equipment>. (accessed: 29.06.25).
- [33] *The Electromagnetic Compatibility Regulations 2016*, UK Government, 2016. [Online]. Available: <https://www.legislation.gov.uk/ukxi/2016/1091/contents>, (accessed: 09.07.25).
- [34] J. Martin, *Rapid application development*, 1991.
- [35] *Computer Misuse Act 1990*, UK Government, 1990. [Online]. Available: <https://www.legislation.gov.uk/ukpga/1990/18/contents>, (accessed: 07.07.25).
- [36] "ESG in PCB: The Risks and Opportunities of ESG for the PCB Industry | Octopart," Octopart. (2023), [Online]. Available: <https://octopart.com/pulse/p/esg-pcb-risks-and-opportunities-esg-pcb-industry>. (accessed: 07.07.25).
- [37] "Certifications," JLCPCB. (2015), [Online]. Available: <https://jlcpcb.com/help/article/Certifications>. (accessed: 07.07.25).
- [38] "Newbury Electronics | PCB Manufacture, Assembly & Design," Newbury Electronics. (2023), [Online]. Available: <https://www.newburyelectronics.co.uk/>. (accessed: 07.07.25).
- [39] "About KiCad | KiCad," KiCad project. (2025), [Online]. Available: <https://www.kicad7org/about/kicad/>. (accessed: 30.06.25).
- [40] "STM32F407VGT6 EDA | CAD 3D Model Download | Digikey," Digikey. (2025), [Online]. Available: <https://www.digikey.co.uk/en/models/2747117>. (accessed: 19.06.25).
- [41] "Free Online PCB CAD Library | Ultra Librarian," Ultra Librarian. (2025), [Online]. Available: <https://www.ultralibrarian.com>. (accessed: 30.06.25).
- [42] "Component Search Engine: Free Symbols, footprints, & 3D models," SupplyFrame. (2025), [Online]. Available: <https://componentsearchengine.com>. (accessed: 30.06.25).
- [43] *Guidelines for oscillator design on STM8AF/AL/S and STM32 MCUs/MPU*, ST Microelectronics, 2025. [Online]. Available: https://www.st.com/resource/en/application_note/an2867-guidelines-for-oscillator-design-on-stm8afals-and-stm32-mcusmpus-stmicroelectronics.pdf, (accessed 19.06.25).
- [44] *Crystal-Oscillators-X50328MSB2GI_C115962*, YXC, 2025. [Online]. Available: https://lcsc.com/datasheet/lcsc_datasheet_2403291504_YXC-Crystal-Oscillators-X50328MSB2GI_C115962.pdf, (accessed: 03.07.25).
- [45] *USB 2.0 Specification*, Universal Serial Bus Implementers Forum, 2025. [Online]. Available: <https://www.usb.org/document-library/usb-20-specification>, (accessed: 29.06.25).
- [46] *TPS20xxB Current-Limited, Power-Distribution Switches*, Texas Instruments, 2025. [Online]. Available: <https://www.ti.com/lit/ds/symlink/tps2052b.pdf>, (accessed: 04.07.25).
- [47] *LMS8117A 1-A Low-Dropout Linear Regulator*, Texas Instruments, 2025. [Online]. Available: https://www.ti.com/lit/ds/symlink/lms8117a.pdf?ts=1751453893469&ref_url=https%253A%252F%252Fwww.google.com%252F, (accessed: 01.07.25).

- [48] *SSSS925800 5.0 (H) mm, 2.0mm-travel Type SSSS9 Series*, Alps Alpine, 2025. [Online]. Available: <https://tech.alpsalpine.com/e/products/detail/SSSS925800/>, (accessed: 04.07.25).
- [49] Paul Falstad. “Circuit Simulator Applet.” (2025), [Online]. Available: <https://www.falstad.com/circuit/>. (accessed: 28.06.25).
- [50] “JLCPCB,” JIALICHUANG. (2025), [Online]. Available: <https://jlcpcb.com/aboutUs>. (accessed: 05.07.25).
- [51] bennymeg. “Fabrication-Toolkit.” (2025), [Online]. Available: <https://github.com/bennymeg/Fabrication-Toolkit>. (accessed: 05.07.25).
- [52] *DSA 03.OME Part 2 (JSP 482) - Defence Code of Practice (DCOP) and Guidance Notes for In-Service and Operational Safety Management of OME*, Ministry of Defence, 2025. [Online]. Available: https://assets.publishing.service.gov.uk/media/5ee8b1a986650c03f0a4304a/DSA_03_OME_PART_2_JSP_482_CH_08_Nov_19_-_mod_gov_uk.pdf, accessed 19.06.25.
- [53] “Onshape,” ptc. (2025), [Online]. Available: <https://www.onshape.com/en/>. (accessed: 05.07.25).
- [54] “ChituBox,” Chitu Systems. (2025), [Online]. Available: <https://www.chitubox.com/en/page/about>. (accessed: 05.07.25).
- [55] “Anycubic Photon,” Anycubic. (2025), [Online]. Available: <https://store.anycubic.com/products/anycubic-photon-3d-printer>. (accessed: 05.07.25).
- [56] “Open On-Chip Debugger.” (2023), [Online]. Available: <https://openocd.org/>. (accessed: 29.06.25).
- [57] “libopencm3,” libopencm3. (2023), [Online]. Available: <https://libopencm3.org/docs/latest/html/index.html>. (accessed: 29.06.25).
- [58] H. Thach. “Tinyusb project,” Adafruit. (2025), [Online]. Available: <https://docs.tinyusb.org/en/latest/>. (accessed: 29.06.25).
- [59] “Stm32cubef4 hal driver mcu component,” STMicroelectronics. (2025), [Online]. Available: <https://github.com/STMicroelectronics/stm32f4xx-hal-driver>. (accessed: 07.07.25).
- [60] “Discovery kit with STM32F407VG MCU,” STMicroelectronics. (2025), [Online]. Available: <https://www.st.com/en/evaluation-tools/stm32f4discovery.html>. (accessed: 07.07.25).
- [61] *dmesg(1)*, 2025. [Online]. Available: <https://www.man7.org/linux/man-pages/man1/dmesg.1.html>, (accessed: 07.07.25).
- [62] M. Drake. “libcyaml: C library for reading and writing YAML.” (2024), [Online]. Available: <https://github.com/tlsa/libcyaml>. (accessed: 08.07.25).
- [63] “ISC License,” Choose a License. (2025), [Online]. Available: <https://choosealicense.com/licenses/isc/>. (accessed: 08.07.25).

Glossary

- 3D** 3 dimension 28
- API** application interface 31
- CAD** computer-aided design 18
- CKAKL** Computer and Keyboard Agnostic Keyboard Layout 8
- DID** device identifier 12, 14, 60
- FS** full speed 11, 20, 21, 35, 58, 59
- HAL** Hardware Access Layer 34
- HS** high speed 11, 21, 35, 59
- I2C** Inter-Integrated Circuit 11, 58
- IC** intergrated circuit 61
- ISC** Internet Software Consortium 61
- ISO** International Organization for Standardization 15
- JSON** JavaScript Object Notation 7, 12, 60
- JTAG** Joint Test Action Group 21, 30, 59
- LED** light emitting diode 2, 3, 5, 7, 24, 29, 31, 32, 34, 37, 62, 63
- LS** low speed 11, 35
- MISO** Master In Slave Out 29
- MOSI** Master Out Slave In 29
- OpenOCD** Open On-Chip Debugger 30
- PCB** Printed Circuit Board 2, 5, 7, 8, 11, 12, 15, 16, 18, 26–28, 30, 31, 33–38, 50–52, 58–61
- RAD** Rapid Application Development 15
- RoHS** Restriction of Hazardous Substances 14, 15, 48
- SPI** Serial Peripheral Interface 3, 5, 7, 11, 23, 26, 29, 33, 34, 59, 64, 66
- STL** Standard Tessellation Language 28
- SWD** Serial Wire Debug 2, 3, 5, 7, 21, 22, 26, 29–31, 38, 59, 61, 62
- UART** Universal Asynchronous Receiver/Transmitter 11, 38, 58
- USB** Universal Serial Bus 2, 5, 8, 11, 14, 15, 20–22, 26, 29, 30, 34–38, 48, 58, 59, 61
- VID** vendor identifier 12, 14, 60
- WEEE** waste electrical and electronic equipment 14–16, 48
- YAML** Yet Another Markup Language 2, 3, 5, 7, 12, 14, 15, 23, 41–43, 48, 51, 59–61, 94, 96

A Editorial Modifications

To provide coherence throughout the document the editor, **JG**, has modified the style, structure and formatting of the text included in the final report. Every effort has been made to not change the meaning of the original text, but this often fails to be the case. This section attempts, in chronological order, to layout the original work provided by each member of the team.

A.1 SH

A.1.1 Paragraph 1

The processing power for the project would be provided by a micro-controller. The primary requirements for the project to function are that it has two USB controllers, is small and has a low power consumption. Research indicated that the maximum amperage was 500 mA in the USB 2.0 standard [11]. As such, a lower power chip which allows for enough overheads for safe operation. In this case an arbitrary limit of half of the maximum power limit will be applied. The table below describes some of the key criteria and an evaluation of the chips researched. 

A.1.2 Paragraph 2

Based on the research regarding USB interfacing, the requirements for interacting with both a computer and keyboard is the ability to interact in both a device and host concurrently. Alternatively, two separate chips could be used which communicate and perform the host and device roles separately. 

A.1.3 Paragraph 3

Due to the limited amount of chips that support two USB instances at once, the range of chips that are usable have been greatly narrowed. As a result, the most likely family of chips that can be used will be the STM32F4XX or FT4232H range. This is due to the support for two USB interfaces concurrently, which will allow for both the keyboard and computer to interact with as little latency as possible, alongside the lack of support for being a host and device concurrently. However, due to the nature of the FT4232H chip, it's sole purpose is to convert USB messages into UART or I2C messages with no further processing. This means that the software required to remap keys couldn't be ran within the chip, making it a less ideal option as it doesn't fit the required use case. Alternatively, the STM32F407VGT6 can fulfil the use case fully and has an adequately small package for a smaller overall form factor. Another factor that benefits the development using the STM32F407VGT6 is access to development boards that can be used out of the box. This will greatly aid in the creation of software and prototyping concepts for the hardware before a PCB is created and manufactured. 

A.1.4 Paragraph 4

The KiCAD electronics design suite [39], an open source and free design software, which supports a multitude of features and plugins was used to create both the schematic and PCB. It did require some custom footprints and symbols to accommodate parts not within the internal KiCAD catalogue [40] [41] [42]. 

A.1.5 Paragraph 5

The first component to be added to the schematic was the STM32F407VGT6, followed by the power and ground pins according to the specification [14, p. 43]. 100 nF decoupling capacitors were added to ensure that power and ground were adequately smoothed, shown in Figure 6. VDDA requires a larger 1 μ F capacitor as the logic is referenced from this voltage. Figure 7 shows the schematic for the power and ground pins. 

A.1.6 Paragraph 6

During this process, a review of the design was carried out with a more experienced design engineer who recommended the addition of 2.2 μ F capacitors for the VCAP pins. The VCAP capacitor values were designated by the power diagram given in Figure 6. 

A.1.7 Paragraph 7

For FS USB to be accessible, an external oscillator was added through following AN2867 [43], which explains the ideal characteristics for the oscillator and the fundamental equations needed to ensure proper operation. To ensure that a wide range of frequencies is achievable, an 8 MHz oscillator was chosen. An internal prescaler can be used to achieve a majority of the frequencies that may be required. The oscillator was connected to dedicated external oscillator pins PH00 and PH01. 

A.1.8 Paragraph 8

One of the essential equations used was the equation for the decoupling capacitors [43, p. 12] and the associated diagram, shown below.

$$C_L = \frac{C_{L1} \times C_{L2}}{C_{L1} + C_{L2}} + C_s \quad (2)$$

Given that $C_L < 20 \text{ pF}$ & $C_s = 5 \text{ [44]}$, $C_{L1} \& C_{L2} = 20 \text{ pF}$. Figure 10, shows the set up for the oscillator. 

A.1.9 Paragraph 9

The STM was connected to its host through the USB HS and FS compatible interface on pins PA11 & PA12 for D- and D+ respectively. This was decided as a larger internal buffer is present for the volumes of data which may need to be transferred to change the configuration. 5 V and ground, via this connector, supply power to both the board and downstream devices. 

A.1.10 Paragraph 10

The host USB data lines were attached to the dedicated USB FS D+ and D- on pins PB15 & PB14 respectively. Ground and 5 V power pass through was attached too. 

A.1.11 Paragraph 11

A review of the design was conducted, there were no resistors for pulling up and down for the host and device lines. This made the design both inoperable and not USB 2.0 compliant. The resistor values used for the data lines are described in the USB specification [45]. Therefore, the D+ line of the device side USB required pulling up to 3.3 V through a $1.5 \text{ k}\Omega$ resistor. Furthermore, the host side would require two $15 \text{ k}\Omega$ resistors for the received data lines. Figure 11 shows the schematic for both hosting and device components. 

A.1.12 Paragraph 12

To allow for programming the STM32, either JTAG or SWD pin header would be required. As such, SWD pins were added due to a smaller connector being required. 3.3 V, SWCLK (PA14), ground, and SWDIO (PB03) were connected to pins 1 to 4 respectively. 

A.1.13 Paragraph 13

A switch is required to change between modes, so that the ability to upload YAML files is toggleable. As such, a high signal must be sent when in one mode and vice versa. The switch decided upon was the SSSS925800 due to its small package and ability to latch in different modes [48]. In conjunction with this, a hardware smoothing low pass filter was added to ensure that bouncing will not occur during cycle. This was achieved through mathematically modelling the filter characteristics and the attenuation of common bouncing frequencies. The full mathematical model can be found in the appendices. This was then tested in Falstad [49], to ensure accuracy. $R = 33$ and $C = 4.7 \times 10^{-6}$ was decided upon as it balances the cutoff point and sharp fall off, which were present in two concurrent indices of the appendix, as such, the middle was chosen between the two. 

A.1.14 Paragraph 14

In order to aid the quick development and debugging of software, a reset pin would be ideal. This will allow for resetting the chip on the fly. This was achieved through a button pulling down the active high signal on the NRST pin of the STM. As per the specification, this requires an extra capacitor to debounce the signal whose value is set by STM. 

A.1.15 Paragraph 15

The netlist created using the schematic was automatically imported into the PCB editor, the footprints were imported from the same sources as Section 6. The components were placed to ensure the thinnest width was accessible. Components were added in the order that they would be connected. Hence, the SWD, current limiting switch and SPI are near the top of the board. The USB data lines were connected via a differential pair, which was length matched to ensure similar impedance. 

A.2 BG

A.2.1 Paragraph 1

A configuration file will be present on the device to enable the user to customise the DID, VID, and keyboard remappings, macros and multimedia keys. To achieve this either YAML or JSON can be used. 

A.2.2 Paragraph 2

YAML is a light-weight, human readable data representation language. Primarily, it is designed to make the format easy to read and understand whilst still providing functionality for complex features. An example YAML file is shown in Listing 2. [25] 

A.2.3 Paragraph 3

JSON is a human readable language which is independent from the programming language being used for the application. It is most commonly used in web-based applications and an example JSON file is shown in Listing 3. [25] 

A.2.4 Paragraph 4

For this device, a YAML configuration file is going to be used due to it requiring less syntax which will make it easier for the user to understand as well as mitigating errors from incorrect syntax. Additionally, comments can be present in YAML files which will allow for extra information to be present to aid the user in configuring the device. Moreover, the “LibCYaml” library can be used for the software implementation of the file parser, reducing the complexity of the software needed for the device, therefore reducing the development time and cost. 

A.2.5 Paragraph 5

On the market there are a wide range of keyboards which all come with varying features. The majority of these features can be put into one of the following categories:

- Remapping of keys which is the process of changing what action or character is associated with a specific key on the keyboard. [26]
- Keyboard macros which allow for a sequence of computer instructions or actions to be mapped to single key press. [27]
- Multimedia keys which are typically present on the majority of keyboards to enable easy control of volume and media playback such as playing and pausing of a video. [28]

The features listed above will be the initial actions that the device will enable for the user. It will allow the user to have all of the keyboard features available no matter what device or physical keyboard is being used. Additionally, all of the features mentioned serve a unified purpose: to increase the productivity of the user. With the ability to take these features wherever the user is working, it is enabling a higher level of productivity no matter the quality of the physical hardware present at the workstation. 

A.2.6 Paragraph 6

Scan codes are generated by a keyboard whenever a user presses a key. These codes do not correlate to the value visually present on the key, instead it refers to the location of the key that is pressed. Shown in Figure 4, are the scan codes for a standard QWERTY keyboard. [29] 

A.2.7 Paragraph 7

For this methodology to work, the project has been split into four major tasks which have been broken up into smaller sub-tasks. These four tasks are: software development; PCB development; deploying software to hardware; and testing.

- Software Development
The software development task is broken down into three sub-tasks which are outlined below.

- USB Stack from Keyboard
The purpose of this stage is to build a USB stack that will interface with a keyboard. This will receive the user input in the form of scan codes as well as enable the usual data communication between the keyboard the device it is connected to, to ensure that the standard keyboard functionality is not altered.
- USB Stack to Host
The purpose of this stage is to build a USB stack that will interface with the host, such as a computer. It will be passed the scan codes from the keyboard and apply any modifications that are specified in the configuration file. These scan codes will then be sent to the connected device.
- YAML File Parser
The purpose of this stage is to build a YAML file parser that will read in the configuration file present. It will parse the data contained within the YAML file into internal data structures that can then be used by other parts of the software, such as the two USB stacks.
- PCB Development
Similarly to the software development task, this task has been broken down into three sub-tasks which are outlined below.
 - Schematic Design
This stage will consist of using KiCad to produce a PCB schematic of a design for the device. This will consist of researching what intergrated circuits (ICs) and electrical components are needed to ensure the device meets requirements and behaves as expected.
 - PCB Design
After the schematic has been designed, KiCad will be used to produce a PCB design. This will involve placing the ICs and electrical components into physical places on a PCB and ensuring the traces, vias, and pads for the ICs do not overlap or interfere with one another.
 - PCB Procurement and Testing
The final stage is to procure this PCB design from a PCB manufacturer. Market research will be conducted to ensure the project is receiving value for money as well as taking into consideration the ethical implications of the manufacturing process. Upon receipt of the PCB, testing will be conducted to ensure the PCB performs as intended as well as ensuring no manufacturing errors are present.
- Deploying Software to Hardware
After all stages of the software and PCB development have been completed, the software will be deployed to the PCB via the use of SWD.
- Acceptance Testing
The final stage of the project will be to conduct testing to verify the device is working as intended and meets the outlined requirements. This will involve testing hardware and software functionality, as well as the joint functionality of these two parts. This will be performed in a logical way to help aid the identification of faults or bugs present, as well as appropriate fixes that need to be applied to the device.



A.2.8 Paragraph 8

To enable the user to configure the device through the use of a YAML file, a parser is required to read the file and store data into appropriate internal structures. The open-source library `libcyaml` [62] provides schema based YAML parsing and serialisation. Additionally, it is licensed under the Internet Software Consortium (ISC) licence [63] which enables the project to use the library freely. 

B Source Code

B.1 Testing

B.1.1 git Log

```
1 * 4ce9f4a - Tue, 8 Jul 2025 08:11:24 +0000 (test_flash)
2 |           Test Flash - committer
3 * e3f9bf3 - Mon, 30 Jun 2025 19:05:44 +0000 (test_spi)
4 |           Test SPI - committer
5 * 03fd609 - Mon, 30 Jun 2025 19:03:40 +0000 (test_serial)
6 |           Test Serial - committer
7 | * 4a665fe - Mon, 30 Jun 2025 19:01:53 +0000 (test_switch)
8 | |           Test Switch - committer
9 | * f99aa91 - Mon, 30 Jun 2025 19:01:23 +0000 (test_led)
10 | /           Test LED - committer
11 * 996376c - Mon, 30 Jun 2025 19:00:17 +0000 (HEAD -> master, test_clock)
12 |           Configure clock - committer
13 * 9a863bd - Mon, 30 Jun 2025 18:51:10 +0000 (test_swd)
14 |           Inital commit - committer
```

Listing 23: Log of test code generated via `git log`.

B.1.2 Test SWD

```
1
2 int main(void)
3 {
4
5     while (1) {
6     }
7 }
```

Listing 24: Code used to test SWD.

B.1.3 Test Clock

```
1
2 #include <libopencm3/stm32/rcc.h>
3
4 int main(void)
5 {
6     rcc_clock_setup_pll(&rcc_hse_8mhz_3v3[RCC_CLOCK_3V3_168MHZ]);
7
8     while (1) {
9     }
10 }
```

Listing 25: Template code with the clock scalars configured for 168 MHz.

B.1.4 Test LED

```
1
2 #include <libopencm3/stm32/rcc.h>
3 #include <libopencm3/stm32/gpio.h>
4
5 #define LED_PIN GPIO2
6
7 int main(void)
8 {
9     rcc_clock_setup_pll(&rcc_hse_8mhz_3v3[RCC_CLOCK_3V3_168MHZ]);
10
11     rcc_periph_clock_enable(RCC_GPIOA);
```

```

12
13     /* configure LED pin */
14     gpio_mode_setup(GPIOA, GPIO_MODE_OUTPUT, GPIO_PUPD_PULLDOWN, LED_PIN);
15
16     while (1) {
17         /* delay */
18         for (volatile unsigned i = 0; i < 0x800000; i++);
19         /* toggle the led */
20         gpio_toggle(GPIOA, LED_PIN);
21     }
22 }

```

Listing 26: Test LED Program.

B.1.5 Test Switch

```

1
2 #include <libopencm3/stm32/rcc.h>
3 #include <libopencm3/stm32/gpio.h>
4
5 #define LED_PIN GPIO2
6 #define SWCH_PIN GPIO3
7
8 int main(void)
9 {
10     rcc_clock_setup_pll(&rcc_hse_8mhz_3v3[RCC_CLOCK_3V3_168MHZ]);
11
12     rcc_periph_clock_enable(RCC_GPIOA);
13
14     /* configure LED pin */
15     gpio_mode_setup(GPIOA, GPIO_MODE_OUTPUT, GPIO_PUPD_PULLDOWN, LED_PIN);
16
17     /* configure switch pin */
18     gpio_mode_setup(GPIOA, GPIO_MODE_INPUT, GPIO_PUPD_NONE, SWCH_PIN);
19
20     while (1) {
21         if (gpio_get(GPIOA, SWCH_PIN)) {
22             /* switch in left position; turn LED on */
23             gpio_clear(GPIOA, LED_PIN);
24         } else {
25             /* switch in right position; turn LED off */
26             gpio_set(GPIOA, LED_PIN);
27         }
28     }
29 }

```

Listing 27: Test Switch Program.

B.1.6 Test Serial

```

1
2 #include <libopencm3/stm32/rcc.h>
3 #include <libopencm3/stm32/gpio.h>
4 #include <libopencm3/stm32/usart.h>
5
6 void usart6_putc(char c);
7 void usart6_puts(char* s);
8
9 /*
10  * Taken from
11  * libopencm3-examples/examples/stm32/f4/stm32f429i-discovery/usart_console/
12  * usart_console.c
13  */
14 void usart6_putc(char c)

```

```

14 {
15     uint32_t reg;
16     do {
17         reg = USART_SR(USART6);
18     } while ((reg & USART_SR_TXE) == 0);
19     USART_DR(USART6) = (uint16_t) c & 0xff;
20 }
21
22 /*
23  * Taken from
24  * libopencm3-examples/examples/stm32/f4/stm32f429i-discovery/usart_console/
25  * usart_console.c
26  */
27 void usart6_puts(char *s)
28 {
29     while (*s != '\000') {
30         usart6_putc(*s);
31         /* Add in a carriage return, after sending line feed */
32         if (*s == '\n') {
33             usart6_putc('\r');
34         }
35         s++;
36     }
37 }
38
39 int main(void)
40 {
41     rcc_clock_setup_pll(&rcc_hse_8mhz_3v3[RCC_CLOCK_3V3_168MHZ]);
42
43     rcc_periph_clock_enable(RCC_GPIOA);
44     rcc_periph_clock_enable(RCC_GPIOC);
45     rcc_periph_clock_enable(RCC_USART6);
46
47     /* Configure USART6 pins */
48     gpio_mode_setup(GPIOC, GPIO_MODE_AF, GPIO_PUPD_NONE, GPIO6 | GPIO7);
49     gpio_set_af(GPIOC, GPIO_AF8, GPIO6 | GPIO7);
50
51     /* Configure USART6 */
52     usart_set_baudrate(USART6, 115200);
53     usart_set_databits(USART6, 8);
54     usart_set_stopbits(USART6, USART_STOPBITS_1);
55     usart_set_mode(USART6, USART_MODE_TX);
56     usart_set_parity(USART6, USART_PARITY_NONE);
57     usart_set_flow_control(USART6, USART_FLOWCONTROL_NONE);
58     usart_enable(USART6);
59
60     usart6_puts("Hello World!\r\n");
61
62     while (1) {
63     }
64 }

```

Listing 28: Test Serial Program.

B.1.7 Test SPI

```

1 #include <stdio.h>
2
3 #include <libopencm3/stm32/rcc.h>
4 #include <libopencm3/stm32/gpio.h>
5 #include <libopencm3/stm32/usart.h>
6 #include <libopencm3/stm32/spi.h>
7
8 void usart6_putc(char c);
9 void usart6_puts(char* s);

```

```

10
11 /*
12  * Taken from
13  * libopencm3-examples/examples/stm32/f4/stm32f429i-discovery/usart_console/
14  * usart_console.c
15  */
16 void usart6_putc(char c)
17 {
18     uint32_t reg;
19     do {
20         reg = USART_SR(USART6);
21     } while ((reg & USART_SR_TXE) == 0);
22     USART_DR(USART6) = (uint16_t) c & 0xff;
23 }
24
25 /*
26  * Taken from
27  * libopencm3-examples/examples/stm32/f4/stm32f429i-discovery/usart_console/
28  * usart_console.c
29  */
30 void usart6_puts(char *s)
31 {
32     while (*s != '\000') {
33         usart6_putc(*s);
34         /* Add in a carriage return, after sending line feed */
35         if (*s == '\n') {
36             usart6_putc('\r');
37         }
38         s++;
39     }
40 }
41
42 uint16_t read_man_dev_id()
43 {
44     gpio_clear(GPIOA, GPIO4);
45     spi_send(SPI1, 0x90);
46     spi_read(SPI1);
47     for (unsigned i = 0; i < 3; i++) {
48         spi_send(SPI1, 0);
49         spi_read(SPI1);
50     }
51     spi_send(SPI1, 0);
52     /* Manufacture ID */
53     uint16_t out = spi_read(SPI1);
54     spi_send(SPI1, 0);
55     /* Device ID */
56     out = out << 8 | spi_read(SPI1);
57     gpio_set(GPIOA, GPIO4);
58     return out;
59 }
60
61 int main(void)
62 {
63     rcc_clock_setup_pll(&rcc_hse_8mhz_3v3[RCC_CLOCK_3V3_168MHZ]);
64
65     rcc_periph_clock_enable(RCC_GPIOA);
66     rcc_periph_clock_enable(RCC_GPIOC);
67     rcc_periph_clock_enable(RCC_USART6);
68     rcc_periph_clock_enable(RCC_SPI1);
69
70     /* Configure USART6 pins */
71     gpio_mode_setup(GPIOC, GPIO_MODE_AF, GPIO_PUPD_NONE, GPIO6 | GPIO7);
72     gpio_set_af(GPIOC, GPIO_AF8, GPIO6 | GPIO7);
73
74     /* Configure USART6 */

```

```

74     usart_set_baudrate(USART6, 115200);
75     usart_set_databits(USART6, 8);
76     usart_set_stopbits(USART6, USART_STOPBITS_1);
77     usart_set_mode(USART6, USART_MODE_TX);
78     usart_set_parity(USART6, USART_PARITY_NONE);
79     usart_set_flow_control(USART6, USART_FLOWCONTROL_NONE);
80     usart_enable(USART6);
81
82     /* Configure SPI1 */
83     gpio_mode_setup(GPIOA, GPIO_MODE_AF, GPIO_PUPD_PULLDOWN,
84                     GPIO5 | GPIO6 | GPIO7);
85     gpio_set_af(GPIOA, GPIO_AF5, GPIO5 | GPIO6 | GPIO7);
86     gpio_set_output_options(GPIOA, GPIO_OTYPE_PP, GPIO_OSPEED_25MHZ,
87                             GPIO5 | GPIO7);
88
89     /* Chip select line */
90     gpio_set(GPIOA, GPIO4);
91     gpio_mode_setup(GPIOA, GPIO_MODE_OUTPUT, GPIO_PUPD_NONE, GPIO4);
92
93     uint32_t cr_tmp = SPI_CR1_BAUDRATE_FPCLK_DIV_8 |
94                       SPI_CR1_MSTR |
95                       SPI_CR1_SPE |
96                       SPI_CR1_CPHA |
97                       SPI_CR1_CPOL_CLK_TO_1_WHEN_IDLE;
98
99     SPI_CR2(SPI1) |= SPI_CR2_SSOE;
100    SPI_CR1(SPI1) = cr_tmp;
101
102    char buf[3];
103    uint16_t ids = read_man_dev_id();
104    snprintf(buf, sizeof(buf), "%02x", ids >> 8);
105    usart6_puts("Manufacture_ID: 0x");
106    usart6_puts(buf);
107    usart6_puts("\r\n");
108    snprintf(buf, sizeof(buf), "%02x", ids & 0xff);
109    usart6_puts("Device_ID: 0x");
110    usart6_puts(buf);
111    usart6_puts("\r\n");
112
113    while (1) {
114    }
115 }

```

Listing 29: Test SPI Program.

B.1.8 Test Flash

```

1  #include <stdio.h>
2
3  #include <libopenm3/stm32/rcc.h>
4  #include <libopenm3/stm32/gpio.h>
5  #include <libopenm3/stm32/usart.h>
6  #include <libopenm3/stm32/spi.h>
7
8  void usart6_putc(char c);
9  void usart6_puts(char* s);
10
11  /*
12   * Taken from
13   * libopenm3-examples/examples/stm32/f4/stm32f429i-discovery/usart_console/
14   * usart_console.c
15   */
16  void usart6_putc(char c)
17  {
18      uint32_t reg;

```

```

18         do {
19             reg = USART_SR(USART6);
20         } while ((reg & USART_SR_TXE) == 0);
21         USART_DR(USART6) = (uint16_t) c & 0xff;
22     }
23
24     /*
25     * Taken from
26     * libopenocm3-examples/examples/stm32/f4/stm32f429i-discovery/usart_console/
27     * usart_console.c
28     */
29     void usart6_puts(char *s)
30     {
31         while (*s != '\000') {
32             usart6_putc(*s);
33             /* Add in a carriage return, after sending line feed */
34             if (*s == '\n') {
35                 usart6_putc('\r');
36             }
37             s++;
38         }
39     }
40
41     void write_enable(void);
42     uint8_t read_status_register_1(void);
43     void page_program(uint32_t addr, uint8_t len, uint8_t* arr);
44     void read_spi(uint32_t addr, uint8_t len, uint8_t* arr);
45     int write_spi(uint32_t addr, uint8_t len, uint8_t* arr);
46
47     void write_enable(void)
48     {
49         gpio_clear(GPIOA, GPIO4);
50
51         spi_send(SPI1, 0x06);
52         spi_read(SPI1);
53
54         gpio_set(GPIOA, GPIO4);
55     }
56
57     uint8_t read_status_register_1(void)
58     {
59         gpio_clear(GPIOA, GPIO4);
60
61         spi_send(SPI1, 0x05);
62         spi_read(SPI1);
63
64         spi_send(SPI1, 0);
65         uint8_t out = spi_read(SPI1);
66
67         gpio_set(GPIOA, GPIO4);
68         return out;
69     }
70
71     void page_program(uint32_t addr, uint8_t len, uint8_t* arr)
72     {
73         gpio_clear(GPIOA, GPIO4);
74
75         spi_send(SPI1, 0x02);
76         spi_read(SPI1);
77
78         spi_send(SPI1, addr >> 16);
79         spi_read(SPI1);
80         spi_send(SPI1, addr >> 8);
81         spi_read(SPI1);
82         spi_send(SPI1, addr);
83         spi_read(SPI1);

```

```

83
84     for (uint8_t i = 0; i < len; i++){
85         spi_send(SPI1, arr[i]);
86         spi_read(SPI1);
87     }
88
89     gpio_set(GPIOA, GPIO4);
90 }
91
92 int write_spi(uint32_t addr, uint8_t len, uint8_t* arr)
93 {
94     write_enable();
95     uint8_t status = read_status_register_1();
96     if (!(status & 0x02)){
97         return -1;
98     }
99     page_program(addr, len, arr);
100    while(read_status_register_1() & 0x01);
101    return 0;
102 }
103
104 void read_spi(uint32_t addr, uint8_t len, uint8_t* arr)
105 {
106     gpio_clear(GPIOA, GPIO4);
107
108     spi_send(SPI1, 0x03);
109     spi_read(SPI1);
110
111     spi_send(SPI1, addr >> 16);
112     spi_read(SPI1);
113     spi_send(SPI1, addr >> 8);
114     spi_read(SPI1);
115     spi_send(SPI1, addr);
116     spi_read(SPI1);
117
118     for (uint8_t i = 0; i < len; i++){
119         spi_send(SPI1, 0);
120         arr[i] = spi_read(SPI1);
121     }
122
123     gpio_set(GPIOA, GPIO4);
124 }
125
126
127 int main(void)
128 {
129     rcc_clock_setup_pll(&rcc_hse_8mhz_3v3[RCC_CLOCK_3V3_168MHZ]);
130
131     rcc_periph_clock_enable(RCC_GPIOA);
132     rcc_periph_clock_enable(RCC_GPIOC);
133     rcc_periph_clock_enable(RCC_USART6);
134     rcc_periph_clock_enable(RCC_SPI1);
135
136     /* Configure USART6 pins */
137     gpio_mode_setup(GPIOC, GPIO_MODE_AF, GPIO_PUPD_NONE, GPIO6 | GPIO7);
138     gpio_set_af(GPIOC, GPIO_AF8, GPIO6 | GPIO7);
139
140     /* Configure USART6 */
141     usart_set_baudrate(USART6, 115200);
142     usart_set_databits(USART6, 8);
143     usart_set_stopbits(USART6, USART_STOPBITS_1);
144     usart_set_mode(USART6, USART_MODE_TX);
145     usart_set_parity(USART6, USART_PARITY_NONE);
146     usart_set_flow_control(USART6, USART_FLOWCONTROL_NONE);
147     usart_enable(USART6);
148

```

```

149      /* Configure SPI1 */
150      gpio_mode_setup(GPIOA, GPIO_MODE_AF, GPIO_PUPD_PULLDOWN,
151                      GPIO5 | GPIO6 | GPIO7);
152      gpio_set_af(GPIOA, GPIO_AF5, GPIO5 | GPIO6 | GPIO7);
153      gpio_set_output_options(GPIOA, GPIO_OTYPE_PP, GPIO_OSPEED_25MHZ,
154                              GPIO5 | GPIO7);
155
156      /* Chip select line */
157      gpio_set(GPIOA, GPIO4);
158      gpio_mode_setup(GPIOA, GPIO_MODE_OUTPUT, GPIO_PUPD_NONE, GPIO4);
159
160      uint32_t cr_tmp = SPI_CR1_BAUDRATE_FPCLK_DIV_8 |
161                      SPI_CR1_MSTR |
162                      SPI_CR1_SPE |
163                      SPI_CR1_CPHA |
164                      SPI_CR1_CPOL_CLK_TO_1_WHEN_IDLE;
165
166      SPI_CR2(SPI1) |= SPI_CR2_SSOE;
167      SPI_CR1(SPI1) = cr_tmp;
168
169      write_spi(0x0000, 12, "Hello_world");
170      char buf [12];
171      read_spi(0x0000, 12, buf);
172      usart6_puts(buf);
173
174      while (1) {
175      }
176 }

```

Listing 30: Test Flash Program

B.2 tinyusb

B.2.1 git Log

```
1 * 5705908e9 - Tue, 1 Jul 2025 18:47:46 +0000 (HEAD -> master)
2 |       Configure VBUS1 - committer
3 * 78cf2b03c - Tue, 1 Jul 2025 08:15:29 +0000
4 |       Configure USART - committer
5 * 3f2e401e5 - Tue, 1 Jul 2025 08:10:33 +0000
6 |       Configure LED pin - committer
7 * e101516f5 - Tue, 1 Jul 2025 08:01:56 +0000
8 |       Add template for custom board - committer
```

Listing 31: Log of tinyusb code generated via git log.

B.2.2 hw/bsp/stm32f4/boards/stm32f407usb/

```
1 /*
2  * The MIT License (MIT)
3  *
4  * Copyright (c) 2020, Ha Thach (tinyusb.org)
5  *
6  * Permission is hereby granted, free of charge, to any person obtaining a copy
7  * of this software and associated documentation files (the "Software"), to deal
8  * in the Software without restriction, including without limitation the rights
9  * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
10 * copies of the Software, and to permit persons to whom the Software is
11 * furnished to do so, subject to the following conditions:
12 *
13 * The above copyright notice and this permission notice shall be included in
14 * all copies or substantial portions of the Software.
15 *
16 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
17 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
18 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
19 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
20 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
21 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
22 * THE SOFTWARE.
23 *
24 * This file is part of the TinyUSB stack.
25 */
26
27 /* metadata:
28     name: STM32 F407 Discovery
29     url: https://www.st.com/en/evaluation-tools/stm32f4discovery.html
30 */
31
32 #ifndef BOARD_H_
33 #define BOARD_H_
34
35 #ifdef __cplusplus
36     extern "C" {
37 #endif
38
39 // Enable PC6 as the debug log UART
40 #define UART_DEV          USART6
41
42 #define PINID_LED          0
43 #define PINID_UART_TX      1
44 #define PINID_UART_RX      2
45 #define PINID_VBUS1_EN     3
46
47 static board_pindef_t board_pindef[] = {
48     { // LED
```

```

49     .port = GPIOA,
50     .pin_init = { .Pin = GPIO_PIN_2, .Mode = GPIO_MODE_OUTPUT_PP, .Pull =
        GPIO_PULLDOWN, .Speed = GPIO_SPEED_HIGH, .Alternate = 0 },
51     .active_state = 0
52 },
53 { // UART TX
54     .port = GPIOC,
55     .pin_init = { .Pin = GPIO_PIN_6, .Mode = GPIO_MODE_AF_PP, .Pull = GPIO_PULLUP, .
        Speed = GPIO_SPEED_HIGH, .Alternate = GPIO_AF8_USART6 },
56     .active_state = 0
57 },
58 { // UART RX
59     .port = GPIOC,
60     .pin_init = { .Pin = GPIO_PIN_7, .Mode = GPIO_MODE_AF_PP, .Pull = GPIO_PULLUP, .
        Speed = GPIO_SPEED_HIGH, .Alternate = GPIO_AF8_USART6 },
61     .active_state = 0
62 },
63 { // VBUS1 EN
64     .port = GPIOA,
65     .pin_init = { .Pin = GPIO_PIN_1, .Mode = GPIO_MODE_OUTPUT_PP, .Pull = GPIO_NOPULL
        , .Speed = GPIO_SPEED_HIGH, .Alternate = 0 },
66     .active_state = 1
67 }
68 };
69
70 //-----+
71 // RCC Clock
72 //-----+
73 static inline void board_clock_init(void)
74 {
75     RCC_ClkInitTypeDef RCC_ClkInitStruct;
76     RCC_OscInitTypeDef RCC_OscInitStruct;
77
78     /* Enable Power Control clock */
79     __HAL_RCC_PWR_CLK_ENABLE();
80
81     /* The voltage scaling allows optimizing the power consumption when the device is
82        clocked below the maximum system frequency, to update the voltage scaling value
83        regarding system frequency refer to product datasheet. */
84     __HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_VOLTAGE_SCALE1);
85
86     /* Enable HSE Oscillator and activate PLL with HSE as source */
87     RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSE;
88     RCC_OscInitStruct.HSEState = RCC_HSE_ON;
89     RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
90     RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSE;
91     RCC_OscInitStruct.PLL.PLLM = HSE_VALUE/1000000;
92     RCC_OscInitStruct.PLL.PLLN = 336;
93     RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV2;
94     RCC_OscInitStruct.PLL.PLLQ = 7;
95     HAL_RCC_OscConfig(&RCC_OscInitStruct);
96
97     /* Select PLL as system clock source and configure the HCLK, PCLK1 and PCLK2
98        clocks dividers */
99     RCC_ClkInitStruct.ClockType = (RCC_CLOCKTYPE_SYSCLK | RCC_CLOCKTYPE_HCLK |
        RCC_CLOCKTYPE_PCLK1 | RCC_CLOCKTYPE_PCLK2);
100     RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
101     RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
102     RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV4;
103     RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV2;
104     HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_5);
105
106     /* Enable clocks Uart
107     __HAL_RCC_USART6_CLK_ENABLE();
108 }
109

```

```

110 static inline void board_vbus_sense_init(uint8_t rhport) {
111     // vbus sense not supported on this board
112     (void) rhport;
113 }
114
115 static inline void board_vbus_set(uint8_t rhport, bool state) {
116     if (rhport == 1) {
117         board_pindex_t* pindex = &board_pindex[PINID_VBUS1_EN];
118         HAL_GPIO_WritePin(pindex->port, pindex->pin_init.Pin, state == pindex->
            active_state ? GPIO_PIN_SET : GPIO_PIN_RESET);
119     }
120 }
121
122 #ifdef __cplusplus
123 }
124 #endif
125
126 #endif /* BOARD_H_ */

```

Listing 32: tinyusb board.h.

B.2.3 hw/bsp/stm32f4/

```

1  /*
2   * The MIT License (MIT)
3   *
4   * Copyright (c) 2019 Ha Thach (tinyusb.org)
5   *
6   * Permission is hereby granted, free of charge, to any person obtaining a copy
7   * of this software and associated documentation files (the "Software"), to deal
8   * in the Software without restriction, including without limitation the rights
9   * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
10  * copies of the Software, and to permit persons to whom the Software is
11  * furnished to do so, subject to the following conditions:
12  *
13  * The above copyright notice and this permission notice shall be included in
14  * all copies or substantial portions of the Software.
15  *
16  * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
17  * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
18  * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
19  * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
20  * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
21  * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
22  * THE SOFTWARE.
23  *
24  * This file is part of the TinyUSB stack.
25  */
26
27 /* metadata:
28    manufacturer: STMicroelectronics
29 */
30
31 #include "stm32f4xx_hal.h"
32 #include "bsp/board_api.h"
33
34 typedef struct {
35     GPIO_TypeDef* port;
36     GPIO_InitTypeDef pin_init;
37     uint8_t active_state;
38 } board_pindex_t;
39
40 #include "board.h"
41
42 //-----+

```

```

43 // Forward USB interrupt events to TinyUSB IRQ Handler
44 //-----+
45 void OTG_FS_IRQHandler(void) {
46     tusb_int_handler(0, true);
47 }
48
49 void OTG_HS_IRQHandler(void) {
50     tusb_int_handler(1, true);
51 }
52
53 //-----+
54 // MACRO TYPEDEF CONSTANT ENUM
55 //-----+
56 #ifdef UART_DEV
57 UART_HandleTypeDef UartHandle = {
58     .Instance = UART_DEV,
59     .Init = {
60         .BaudRate    = CFG_BOARD_UART_BAUDRATE,
61         .WordLength  = UART_WORDLENGTH_8B,
62         .StopBits    = UART_STOPBITS_1,
63         .Parity       = UART_PARITY_NONE,
64         .HwFlowCtl   = UART_HWCONTROL_NONE,
65         .Mode         = UART_MODE_TX_RX,
66         .OverSampling = UART_OVERSAMPLING_16
67     }
68 };
69 #endif
70
71 void board_init(void) {
72     board_clock_init();
73     //SystemCoreClockUpdate();
74
75     // Enable All GPIOs clocks
76     __HAL_RCC_GPIOA_CLK_ENABLE();
77     __HAL_RCC_GPIOB_CLK_ENABLE();
78     __HAL_RCC_GPIOC_CLK_ENABLE();
79     __HAL_RCC_GPIOD_CLK_ENABLE();
80     #ifdef __HAL_RCC_GPIOE_CLK_ENABLE
81         __HAL_RCC_GPIOE_CLK_ENABLE();
82     #endif
83     #ifdef __HAL_RCC_GPIOF_CLK_ENABLE
84         __HAL_RCC_GPIOF_CLK_ENABLE();
85     #endif
86     #ifdef __HAL_RCC_GPIOG_CLK_ENABLE
87         __HAL_RCC_GPIOG_CLK_ENABLE();
88     #endif
89     __HAL_RCC_GPIOH_CLK_ENABLE();
90     #ifdef __HAL_RCC_GPIOI_CLK_ENABLE
91         __HAL_RCC_GPIOI_CLK_ENABLE();
92     #endif
93     #ifdef __HAL_RCC_GPIOJ_CLK_ENABLE
94         __HAL_RCC_GPIOJ_CLK_ENABLE();
95     #endif
96
97     for (uint8_t i = 0; i < TU_ARRAY_SIZE(board_pindef); i++) {
98         HAL_GPIO_Init(board_pindef[i].port, &board_pindef[i].pin_init);
99     }
100
101     #if CFG_TUSB_OS == OPT_OS_NONE
102         // 1ms tick timer
103         SysTick_Config(SystemCoreClock / 1000);
104     #elif CFG_TUSB_OS == OPT_OS_FREERTOS
105         // Explicitly disable systick to prevent its ISR runs before scheduler start
106         SysTick->CTRL &= ~1U;
107
108         // If freeRTOS is used, IRQ priority is limit by max syscall ( smaller is higher )

```

```

109     NVIC_SetPriority(OTG_FS_IRQn, configLIBRARY_MAX_SYSCALL_INTERRUPT_PRIORITY );
110 #endif
111
112     board_led_write(false);
113
114 #ifdef UART_DEV
115     HAL_UART_Init(&UartHandle);
116 #endif
117
118     //----- USB FS -----//
119     GPIO_InitTypeDef GPIO_InitStructure;
120
121     /* Configure USB D+ D- Pins */
122     GPIO_InitStructure.Pin = GPIO_PIN_11 | GPIO_PIN_12;
123     GPIO_InitStructure.Speed = GPIO_SPEED_HIGH;
124     GPIO_InitStructure.Mode = GPIO_MODE_AF_PP;
125     GPIO_InitStructure.Pull = GPIO_NOPULL;
126     GPIO_InitStructure.Alternate = GPIO_AF10_OTG_FS;
127     HAL_GPIO_Init(GPIOA, &GPIO_InitStructure);
128
129     /* Configure VBUS Pin */
130     GPIO_InitStructure.Pin = GPIO_PIN_9;
131     GPIO_InitStructure.Mode = GPIO_MODE_INPUT;
132     GPIO_InitStructure.Pull = GPIO_NOPULL;
133     HAL_GPIO_Init(GPIOA, &GPIO_InitStructure);
134
135     /* ID Pin */
136     GPIO_InitStructure.Pin = GPIO_PIN_10;
137     GPIO_InitStructure.Mode = GPIO_MODE_AF_OD;
138     GPIO_InitStructure.Pull = GPIO_PULLUP;
139     GPIO_InitStructure.Speed = GPIO_SPEED_HIGH;
140     GPIO_InitStructure.Alternate = GPIO_AF10_OTG_FS;
141     HAL_GPIO_Init(GPIOA, &GPIO_InitStructure);
142
143     // Enable USB OTG clock
144     __HAL_RCC_USB_OTG_FS_CLK_ENABLE();
145
146     //----- USB HS -----//
147 #ifndef __HAL_RCC_USB_OTG_HS_CLK_ENABLE
148     GPIO_InitStructure.Pin = GPIO_PIN_14 | GPIO_PIN_15;
149     GPIO_InitStructure.Speed = GPIO_SPEED_HIGH;
150     GPIO_InitStructure.Mode = GPIO_MODE_AF_PP;
151     GPIO_InitStructure.Pull = GPIO_NOPULL;
152     GPIO_InitStructure.Alternate = GPIO_AF12_OTG_HS_FS;
153     HAL_GPIO_Init(GPIOB, &GPIO_InitStructure);
154
155     /* Configure VBUS Pin */
156     GPIO_InitStructure.Pin = GPIO_PIN_13;
157     GPIO_InitStructure.Mode = GPIO_MODE_INPUT;
158     GPIO_InitStructure.Pull = GPIO_NOPULL;
159     HAL_GPIO_Init(GPIOB, &GPIO_InitStructure);
160
161     /* ID Pin */
162     GPIO_InitStructure.Pin = GPIO_PIN_12;
163     GPIO_InitStructure.Mode = GPIO_MODE_AF_OD;
164     GPIO_InitStructure.Pull = GPIO_PULLUP;
165     GPIO_InitStructure.Speed = GPIO_SPEED_HIGH;
166     GPIO_InitStructure.Alternate = GPIO_AF12_OTG_HS_FS;
167     HAL_GPIO_Init(GPIOB, &GPIO_InitStructure);
168
169     // Enable USB OTG clock
170     __HAL_RCC_USB_OTG_HS_CLK_ENABLE();
171 #endif
172
173 #ifdef STM32F412Zx
174     /* Configure POWER_SWITCH IO pin */

```

```

175     __HAL_RCC_GPIOG_CLK_ENABLE();
176     GPIO_InitStruct.Pin = GPIO_PIN_8;
177     GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_OD;
178     GPIO_InitStruct.Pull = GPIO_NOPULL;
179     HAL_GPIO_Init(GPIOG, &GPIO_InitStruct);
180 #endif
181
182 #if CFG_TUD_ENABLED
183     board_vbus_sense_init(BOARD_TUD_RHPORT);
184 #endif
185
186 #if CFG_TUH_ENABLED
187     board_vbus_set(BOARD_TUH_RHPORT, true);
188 #endif
189 }
190
191 //-----+
192 // Board porting API
193 //-----+
194
195 void board_led_write(bool state) {
196 #ifndef PINID_LED
197     board_pindef_t* pindef = &board_pindef[PINID_LED];
198     GPIO_PinState pin_state = state == pindef->active_state ? GPIO_PIN_SET :
199         GPIO_PIN_RESET;
200     HAL_GPIO_WritePin(pindef->port, pindef->pin_init.Pin, pin_state);
201 #else
202     (void) state;
203 #endif
204 }
205
206 uint32_t board_button_read(void) {
207 #ifndef PINID_BUTTON
208     board_pindef_t* pindef = &board_pindef[PINID_BUTTON];
209     return pindef->active_state == HAL_GPIO_ReadPin(pindef->port, pindef->pin_init.Pin)
210         ;
211 #else
212     return 0;
213 #endif
214 }
215
216 size_t board_get_unique_id(uint8_t id[], size_t max_len) {
217     (void) max_len;
218     volatile uint32_t *stm32_uuid = (volatile uint32_t *) UID_BASE;
219     uint32_t *id32 = (uint32_t *) (uintptr_t) id;
220     uint8_t const len = 12;
221
222     id32[0] = stm32_uuid[0];
223     id32[1] = stm32_uuid[1];
224     id32[2] = stm32_uuid[2];
225
226     return len;
227 }
228
229 int board_uart_read(uint8_t *buf, int len) {
230     (void) buf;
231     (void) len;
232     return 0;
233 }
234
235 int board_uart_write(void const *buf, int len) {
236 #ifndef UART_DEV
237     HAL_UART_Transmit(&UartHandle, (uint8_t *) (uintptr_t) buf, len, 0xffff);
238     return len;
239 #else
240     (void) buf; (void) len;
241 #endif
242 }

```

```

239     return 0;
240 #endif
241 }
242
243 #if CFG_TUSB_OS == OPT_OS_NONE
244 volatile uint32_t system_ticks = 0;
245
246 void SysTick_Handler(void) {
247     HAL_IncTick();
248     system_ticks++;
249 }
250
251 uint32_t board_millis(void) {
252     return system_ticks;
253 }
254
255 #endif
256
257 void HardFault_Handler(void) {
258     __asm("BKPT_#0\n");
259 }
260
261 // Required by __libc_init_array in startup code if we are compiling using
262 // -nostdlib/-nostartfiles.
263 void _init(void) {
264 }

```

Listing 33: tinyusb family.c.

B.3 Remapper

B.3.1 git Log

```
1 * 448f4b0 - Sun, 6 Jul 2025 06:27:58 +0000 (HEAD -> master)
2 |           Add remapping - committer
3 * 1cd908f - Fri, 4 Jul 2025 12:45:52 +0000
4 |           Update device name - committer
5 * 1064f5a - Thu, 3 Jul 2025 19:34:50 +0000
6 |           Wire - committer
7 * 96648e9 - Wed, 2 Jul 2025 07:36:14 +0000 (test_device_host)
8 |           USB Host and Device work simultaneously - committer
9 * 8183f0b - Tue, 1 Jul 2025 19:01:22 +0000 (test_host)
10 |          USB Host works - committer
11 * 665aaf8 - Tue, 1 Jul 2025 13:15:28 +0000
12 |          Update build system - committer
13 * 2868921 - Tue, 1 Jul 2025 12:57:11 +0000 (test_device)
14 |          USART works - committer
15 * fe9be48 - Tue, 1 Jul 2025 08:13:26 +0000
16 |          LED blinks - committer
17 * 1fd57d9 - Tue, 1 Jul 2025 08:06:45 +0000
18 |          Update target - committer
19 * 4bcd0f2 - Tue, 1 Jul 2025 08:05:35 +0000
20 |          Intial successful build - committer
```

Listing 34: Log of remapper code generated via git log.

B.3.2 src

```
1 /*
2  * The MIT License (MIT)
3  *
4  * Copyright (c) 2019 Ha Thach (tinyusb.org)
5  *
6  * Permission is hereby granted, free of charge, to any person obtaining a copy
7  * of this software and associated documentation files (the "Software"), to deal
8  * in the Software without restriction, including without limitation the rights
9  * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
10 * copies of the Software, and to permit persons to whom the Software is
11 * furnished to do so, subject to the following conditions:
12 *
13 * The above copyright notice and this permission notice shall be included in
14 * all copies or substantial portions of the Software.
15 *
16 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
17 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
18 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
19 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
20 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
21 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
22 * THE SOFTWARE.
23 *
24 */
25
26 #include <stdlib.h>
27
28 #include "bsp/board_api.h"
29 #include "tusb.h"
30
31 void led_blinking_task(void);
32
33 int main(void) {
34     board_init();
35
36     // init host stack on configured roothub port
```

```

37     tusb_rhport_init_t host_init = {
38         .role = TUSB_ROLE_HOST,
39         .speed = TUSB_SPEED_AUTO
40     };
41     tusb_init(BOARD_TUH_RHPORT, &host_init);
42
43     // init device stack on configured roothub port
44     tusb_rhport_init_t dev_init = {
45         .role = TUSB_ROLE_DEVICE,
46         .speed = TUSB_SPEED_AUTO
47     };
48     tusb_init(BOARD_TUD_RHPORT, &dev_init);
49
50     if (board_init_after_tusb) {
51         board_init_after_tusb();
52     }
53
54     while (1) {
55         // tinyusb host task
56         tuh_task();
57         // tinyusb device task
58         tud_task();
59
60         led_blinking_task();
61     }
62 }
63
64 //-----+
65 // TinyUSB Callbacks
66 //-----+
67
68 void tuh_mount_cb(uint8_t dev_addr) {
69     // application set-up
70     printf("A device with address %u is mounted\r\n", dev_addr);
71 }
72
73 void tuh_umount_cb(uint8_t dev_addr) {
74     // application tear-down
75     printf("A device with address %u is unmounted\r\n", dev_addr);
76 }
77
78
79 //-----+
80 // Blinking Task
81 //-----+
82 void led_blinking_task(void) {
83     const uint32_t interval_ms = 1000;
84     static uint32_t start_ms = 0;
85
86     static bool led_state = false;
87
88     // Blink every interval ms
89     if (board_millis() - start_ms < interval_ms) {
90         return; // not enough time
91     }
92     start_ms += interval_ms;
93
94     board_led_write(led_state);
95     led_state = 1 - led_state; // toggle
96 }

```

Listing 35: Remapper main.c.

```

1  /*
2   * The MIT License (MIT)
3   *

```

```

4  * Copyright (c) 2021, Ha Thach (tinyusb.org)
5  *
6  * Permission is hereby granted, free of charge, to any person obtaining a copy
7  * of this software and associated documentation files (the "Software"), to deal
8  * in the Software without restriction, including without limitation the rights
9  * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
10 * copies of the Software, and to permit persons to whom the Software is
11 * furnished to do so, subject to the following conditions:
12 *
13 * The above copyright notice and this permission notice shall be included in
14 * all copies or substantial portions of the Software.
15 *
16 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
17 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
18 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
19 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
20 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
21 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
22 * THE SOFTWARE.
23 *
24 */
25
26 #include "bsp/board_api.h"
27 #include "tusb.h"
28
29 //-----+
30 // MACRO TYPEDEF CONSTANT ENUM DECLARATION
31 //-----+
32 #define MAX_REPORT 4
33
34 // Each HID instance can has multiple reports
35 static struct {
36     uint8_t report_count;
37     tuh_hid_report_info_t report_info[MAX_REPORT];
38 } hid_info[CFG_TUH_HID];
39
40 static void process_kbd_report(hid_keyboard_report_t const *report);
41 static void process_mouse_report(hid_mouse_report_t const *report);
42 static void process_generic_report(uint8_t dev_addr, uint8_t instance, uint8_t const
    *report, uint16_t len);
43
44 //-----+
45 // TinyUSB Callbacks
46 //-----+
47
48 // Invoked when device with hid interface is mounted
49 // Report descriptor is also available for use. tuh_hid_parse_report_descriptor()
50 // can be used to parse common/simple enough descriptor.
51 // Note: if report descriptor length > CFG_TUH_ENUMERATION_BUFSIZE, it will be
    skipped
52 // therefore report_desc = NULL, desc_len = 0
53 void tuh_hid_mount_cb(uint8_t dev_addr, uint8_t instance, uint8_t const *desc_report,
    uint16_t desc_len) {
54     printf("HID_device_address=%d,instance=%d,is mounted\r\n", dev_addr, instance)
        ;
55
56     // Interface protocol (hid_interface_protocol_enum_t)
57     const char *protocol_str[] = {"None", "Keyboard", "Mouse"};
58     uint8_t const itf_protocol = tuh_hid_interface_protocol(dev_addr, instance);
59
60     printf("HID_Interface_Protocol=%s\r\n", protocol_str[itf_protocol]);
61
62     // By default, host stack will use boot protocol on supported interface.
63     // Therefore for this simple example, we only need to parse generic report
        descriptor (with built-in parser)
64     if (itf_protocol == HID_ITF_PROTOCOL_NONE) {

```

```

65     hid_info[instance].report_count = tuh_hid_parse_report_descriptor(hid_info[
        instance].report_info, MAX_REPORT, desc_report, desc_len);
66     printf("HID has %u reports\n", hid_info[instance].report_count);
67 }
68
69 // request to receive report
70 // tuh_hid_report_received_cb() will be invoked when report is available
71 if (!tuh_hid_receive_report(dev_addr, instance)) {
72     printf("Error: cannot request to receive report\n");
73 }
74 }
75
76 // Invoked when device with hid interface is un-mounted
77 void tuh_hid_umount_cb(uint8_t dev_addr, uint8_t instance) {
78     printf("HID device address = %d, instance = %d is unmounted\n", dev_addr,
        instance);
79 }
80
81 extern void send_kbd_report(hid_keyboard_report_t const *report);
82
83 // Invoked when received report from device via interrupt endpoint
84 void tuh_hid_report_received_cb(uint8_t dev_addr, uint8_t instance, uint8_t const *
    report, uint16_t len) {
85     uint8_t const itf_protocol = tuh_hid_interface_protocol(dev_addr, instance);
86
87     switch (itf_protocol) {
88     case HID_ITF_PROTOCOL_KEYBOARD:
89         TU_LOG2("HID receive boot keyboard report\n");
90         process_kbd_report((hid_keyboard_report_t const *) report);
91         break;
92
93     case HID_ITF_PROTOCOL_MOUSE:
94         TU_LOG2("HID receive boot mouse report\n");
95         process_mouse_report((hid_mouse_report_t const *) report);
96         break;
97
98     default:
99         // Generic report requires matching ReportID and contents with previous parsed
        report info
100         process_generic_report(dev_addr, instance, report, len);
101         break;
102     }
103
104     // continue to request to receive report
105     if (!tuh_hid_receive_report(dev_addr, instance)) {
106         printf("Error: cannot request to receive report\n");
107     }
108 }
109
110 //-----+
111 // Keyboard
112 //-----+
113
114 static void process_kbd_report(hid_keyboard_report_t const *report) {
115     send_kbd_report(report);
116 }
117
118 //-----+
119 // Mouse
120 //-----+
121
122 static void process_mouse_report(hid_mouse_report_t const *report) {
123     (void) report;
124 }
125
126 //-----+

```

```

127 // Generic Report
128 //-----+
129 static void process_generic_report(uint8_t dev_addr, uint8_t instance, uint8_t const
    *report, uint16_t len) {
130     (void) dev_addr;
131     (void) len;
132
133     uint8_t const rpt_count = hid_info[instance].report_count;
134     tuh_hid_report_info_t *rpt_info_arr = hid_info[instance].report_info;
135     tuh_hid_report_info_t *rpt_info = NULL;
136
137     if (rpt_count == 1 && rpt_info_arr[0].report_id == 0) {
138         // Simple report without report ID as 1st byte
139         rpt_info = &rpt_info_arr[0];
140     } else {
141         // Composite report, 1st byte is report ID, data starts from 2nd byte
142         uint8_t const rpt_id = report[0];
143
144         // Find report id in the array
145         for (uint8_t i = 0; i < rpt_count; i++) {
146             if (rpt_id == rpt_info_arr[i].report_id) {
147                 rpt_info = &rpt_info_arr[i];
148                 break;
149             }
150         }
151
152         report++;
153         len--;
154     }
155
156     if (!rpt_info) {
157         printf("Couldn't find report info!\r\n");
158         return;
159     }
160
161     // For complete list of Usage Page & Usage checkout src/class/hid/hid.h. For
    examples:
162     // - Keyboard : Desktop, Keyboard
163     // - Mouse : Desktop, Mouse
164     // - Gamepad : Desktop, Gamepad
165     // - Consumer Control (Media Key) : Consumer, Consumer Control
166     // - System Control (Power key) : Desktop, System Control
167     // - Generic (vendor) : 0xFFxx, xx
168     if (rpt_info->usage_page == HID_USAGE_PAGE_DESKTOP) {
169         switch (rpt_info->usage) {
170             case HID_USAGE_DESKTOP_KEYBOARD:
171                 TU_LOG2("HID receive keyboard report\r\n");
172                 // Assume keyboard follow boot report layout
173                 process_kbd_report((hid_keyboard_report_t const *) report);
174                 break;
175
176             case HID_USAGE_DESKTOP_MOUSE:
177                 TU_LOG2("HID receive mouse report\r\n");
178                 // Assume mouse follow boot report layout
179                 process_mouse_report((hid_mouse_report_t const *) report);
180                 break;
181
182             default:
183                 printf("report[%u]\n", rpt_info->report_id);
184                 for (uint8_t i = 0; i < len; i++) {
185                     printf("%02X\n", report[i]);
186                 }
187                 printf("\r\n");
188                 break;
189         }
190     }

```

Listing 36: Remapper hid_host.c.

```

1  /*
2   * The MIT License (MIT)
3   *
4   * Copyright (c) 2019 Ha Thach (tinyusb.org)
5   *
6   * Permission is hereby granted, free of charge, to any person obtaining a copy
7   * of this software and associated documentation files (the "Software"), to deal
8   * in the Software without restriction, including without limitation the rights
9   * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
10  * copies of the Software, and to permit persons to whom the Software is
11  * furnished to do so, subject to the following conditions:
12  *
13  * The above copyright notice and this permission notice shall be included in
14  * all copies or substantial portions of the Software.
15  *
16  * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
17  * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
18  * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
19  * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
20  * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
21  * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
22  * THE SOFTWARE.
23  *
24  */
25
26 #include "bsp/board_api.h"
27 #include "tusb.h"
28
29 #include "usb_descriptors.h"
30 #include "kbd_remap.h"
31
32 // Invoked when usb bus is suspended
33 // remote_wakeup_en : if host allow us to perform remote wakeup
34 // Within 7ms, device must draw an average of current less than 2.5 mA from bus
35 void tud_suspend_cb(bool remote_wakeup_en)
36 {
37     (void) remote_wakeup_en;
38 }
39
40 static hid_keyboard_report_t const *to_send = NULL;
41
42 // Invoked when usb bus is resumed
43 void tud_resume_cb(void)
44 {
45     if (to_send != NULL) {
46         tud_hid_keyboard_report(REPORT_ID_KEYBOARD, to_send->modifier, to_send->keycode);
47         to_send = NULL;
48     }
49 }
50
51 //-----+
52 // USB HID
53 //-----+
54
55 void send_kbd_report(hid_keyboard_report_t const *report)
56 {
57     // skip if hid is not ready yet
58     if ( !tud_hid_ready() ) return;
59
60     // Remote wakeup
61     if ( tud_suspended() )
62     {

```

```

63     to_send = report;
64     // Wake up host if we are in suspend mode
65     // and REMOTE_WAKEUP feature is enabled by host
66     tud_remote_wakeup();
67 }else
68 {
69     uint8_t keycodes[] = {0,0,0,0,0,0};
70     for (unsigned i = 0; i < 6; i++) {
71         keycodes[i] = kbd_remaps[report->keycode[i]];
72     }
73     tud_hid_keyboard_report(0, report->modifier, keycodes);
74 }
75 }
76
77 // Invoked when sent REPORT successfully to host
78 // Application can use this to send the next report
79 // Note: For composite reports, report[0] is report ID
80 void tud_hid_report_complete_cb(uint8_t instance, uint8_t const* report, uint16_t len
81 )
82 {
83     (void) instance; (void) report; (void) len;
84 }
85
86 // Invoked when received GET_REPORT control request
87 // Application must fill buffer report's content and return its length.
88 // Return zero will cause the stack to STALL request
89 uint16_t tud_hid_get_report_cb(uint8_t instance, uint8_t report_id, hid_report_type_t
90     report_type, uint8_t* buffer, uint16_t reqlen)
91 {
92     // TODO not Implemented
93     (void) instance;
94     (void) report_id;
95     (void) report_type;
96     (void) buffer;
97     (void) reqlen;
98
99     return 0;
100 }
101
102 // Invoked when received SET_REPORT control request or
103 // received data on OUT endpoint ( Report ID = 0, Type = 0 )
104 void tud_hid_set_report_cb(uint8_t instance, uint8_t report_id, hid_report_type_t
105     report_type, uint8_t const* buffer, uint16_t bufsize)
106 {
107     (void) instance; (void) report_id;
108
109     (void) buffer; (void) bufsize;
110
111     if (report_type == HID_REPORT_TYPE_OUTPUT)
112     {
113     }
114 }

```

Listing 37: Remapper hid_device.c.

```

1
2 #ifndef _KBD_REMAP_H_
3 /* _KBD_REMAP_H_ */
4
5 struct kbd_remap {
6     hid_keyboard_modifier_bm_t modifier;
7     uint8_t keycode;
8 };
9
10 #define RESERVED() 0
11

```

```

12  const uint8_t kbd_remaps[] = {
13      /* HID_KEY_NONE */
14
15      /* HID_KEY_A */
16      /* HID_KEY_B */
17      /* HID_KEY_C */
18      /* HID_KEY_D */
19      /* HID_KEY_E */
20      /* HID_KEY_F */
21      /* HID_KEY_G */
22      /* HID_KEY_H */
23      /* HID_KEY_I */
24      /* HID_KEY_J */
25      /* HID_KEY_K */
26      /* HID_KEY_L */
27      /* HID_KEY_M */
28      /* HID_KEY_N */
29      /* HID_KEY_O */
30      /* HID_KEY_P */
31      /* HID_KEY_Q */
32      /* HID_KEY_R */
33      /* HID_KEY_S */
34      /* HID_KEY_T */
35      /* HID_KEY_U */
36      /* HID_KEY_V */
37      /* HID_KEY_W */
38      /* HID_KEY_X */
39      /* HID_KEY_Y */
40      /* HID_KEY_Z */
41      /* HID_KEY_1 */
42      /* HID_KEY_2 */
43      /* HID_KEY_3 */
44      /* HID_KEY_4 */
45      /* HID_KEY_5 */
46      /* HID_KEY_6 */
47      /* HID_KEY_7 */
48      /* HID_KEY_8 */
49      /* HID_KEY_9 */
50      /* HID_KEY_0 */
51      /* HID_KEY_ENTER */
52      /* HID_KEY_ESCAPE */
53      /* HID_KEY_BACKSPACE */
54      /* HID_KEY_TAB */
55      /* HID_KEY_SPACE */
56      /* HID_KEY_MINUS */
57      /* HID_KEY_EQUAL */
58      /* HID_KEY_BRACKET_LEFT */
59      /* HID_KEY_BRACKET_RIGHT */
60      /* HID_KEY_BACKSLASH */
61      /* HID_KEY_EUROPE_1 */
62      /* HID_KEY_SEMICOLON */
63      /* HID_KEY_APOSTROPHE */
64      /* HID_KEY_GRAVE */
65      /* HID_KEY_COMMA */
66      /* HID_KEY_PERIOD */
67      /* HID_KEY_SLASH */
68      /* HID_KEY_CAPS_LOCK */
69      /* HID_KEY_F1 */
70      /* HID_KEY_F2 */
71      /* HID_KEY_F3 */
72      /* HID_KEY_F4 */
73      /* HID_KEY_F5 */
74      /* HID_KEY_F6 */
75      /* HID_KEY_F7 */

```

```

RESERVED(),
RESERVED(),
RESERVED(),
RESERVED(),
HID_KEY_K,
HID_KEY_X,
HID_KEY_V,
HID_KEY_M,
HID_KEY_C,
HID_KEY_N,
HID_KEY_O,
HID_KEY_P,
HID_KEY_H,
HID_KEY_Q,
HID_KEY_R,
HID_KEY_S,
HID_KEY_Z,
HID_KEY_Y,
HID_KEY_I,
HID_KEY_J,
HID_KEY_A,
HID_KEY_D,
HID_KEY_L,
HID_KEY_E,
HID_KEY_G,
HID_KEY_W,
HID_KEY_B,
HID_KEY_U,
HID_KEY_F,
HID_KEY_T,
HID_KEY_1,
HID_KEY_2,
HID_KEY_3,
HID_KEY_4,
HID_KEY_5,
HID_KEY_6,
HID_KEY_7,
HID_KEY_8,
HID_KEY_9,
HID_KEY_0,
HID_KEY_ENTER,
HID_KEY_ESCAPE,
HID_KEY_BACKSPACE,
HID_KEY_TAB,
HID_KEY_SPACE,
HID_KEY_MINUS,
HID_KEY_EQUAL,
HID_KEY_BRACKET_LEFT,
HID_KEY_BRACKET_RIGHT,
HID_KEY_BACKSLASH,
HID_KEY_EUROPE_1,
HID_KEY_SEMICOLON,
HID_KEY_APOSTROPHE,
HID_KEY_GRAVE,
HID_KEY_COMMA,
HID_KEY_PERIOD,
HID_KEY_SLASH,
HID_KEY_CAPS_LOCK,
HID_KEY_F1,
HID_KEY_F2,
HID_KEY_F3,
HID_KEY_F4,
HID_KEY_F5,
HID_KEY_F6,
HID_KEY_F7,

```

78	/* HID_KEY_F8 */	HID_KEY_F8 ,
79	/* HID_KEY_F9 */	HID_KEY_F9 ,
80	/* HID_KEY_F10 */	HID_KEY_F10 ,
81	/* HID_KEY_F11 */	HID_KEY_F11 ,
82	/* HID_KEY_F12 */	HID_KEY_F12 ,
83	/* HID_KEY_PRINT_SCREEN */	HID_KEY_PRINT_SCREEN ,
84	/* HID_KEY_SCROLL_LOCK */	HID_KEY_SCROLL_LOCK ,
85	/* HID_KEY_PAUSE */	HID_KEY_PAUSE ,
86	/* HID_KEY_INSERT */	HID_KEY_INSERT ,
87	/* HID_KEY_HOME */	HID_KEY_HOME ,
88	/* HID_KEY_PAGE_UP */	HID_KEY_PAGE_UP ,
89	/* HID_KEY_DELETE */	HID_KEY_DELETE ,
90	/* HID_KEY_END */	HID_KEY_END ,
91	/* HID_KEY_PAGE_DOWN */	HID_KEY_PAGE_DOWN ,
92	/* HID_KEY_ARROW_RIGHT */	HID_KEY_ARROW_RIGHT ,
93	/* HID_KEY_ARROW_LEFT */	HID_KEY_ARROW_LEFT ,
94	/* HID_KEY_ARROW_DOWN */	HID_KEY_ARROW_DOWN ,
95	/* HID_KEY_ARROW_UP */	HID_KEY_ARROW_UP ,
96	/* HID_KEY_NUM_LOCK */	HID_KEY_NUM_LOCK ,
97	/* HID_KEY_KEYPAD_DIVIDE */	HID_KEY_KEYPAD_DIVIDE ,
98	/* HID_KEY_KEYPAD_MULTIPLY */	HID_KEY_KEYPAD_MULTIPLY ,
99	/* HID_KEY_KEYPAD_SUBTRACT */	HID_KEY_KEYPAD_SUBTRACT ,
100	/* HID_KEY_KEYPAD_ADD */	HID_KEY_KEYPAD_ADD ,
101	/* HID_KEY_KEYPAD_ENTER */	HID_KEY_KEYPAD_ENTER ,
102	/* HID_KEY_KEYPAD_1 */	HID_KEY_KEYPAD_1 ,
103	/* HID_KEY_KEYPAD_2 */	HID_KEY_KEYPAD_2 ,
104	/* HID_KEY_KEYPAD_3 */	HID_KEY_KEYPAD_3 ,
105	/* HID_KEY_KEYPAD_4 */	HID_KEY_KEYPAD_4 ,
106	/* HID_KEY_KEYPAD_5 */	HID_KEY_KEYPAD_5 ,
107	/* HID_KEY_KEYPAD_6 */	HID_KEY_KEYPAD_6 ,
108	/* HID_KEY_KEYPAD_7 */	HID_KEY_KEYPAD_7 ,
109	/* HID_KEY_KEYPAD_8 */	HID_KEY_KEYPAD_8 ,
110	/* HID_KEY_KEYPAD_9 */	HID_KEY_KEYPAD_9 ,
111	/* HID_KEY_KEYPAD_0 */	HID_KEY_KEYPAD_0 ,
112	/* HID_KEY_KEYPAD_DECIMAL */	HID_KEY_KEYPAD_DECIMAL ,
113	/* HID_KEY_EUROPE_2 */	HID_KEY_EUROPE_2 ,
114	/* HID_KEY_APPLICATION */	HID_KEY_APPLICATION ,
115	/* HID_KEY_POWER */	HID_KEY_POWER ,
116	/* HID_KEY_KEYPAD_EQUAL */	HID_KEY_KEYPAD_EQUAL ,
117	/* HID_KEY_F13 */	HID_KEY_F13 ,
118	/* HID_KEY_F14 */	HID_KEY_F14 ,
119	/* HID_KEY_F15 */	HID_KEY_F15 ,
120	/* HID_KEY_F16 */	HID_KEY_F16 ,
121	/* HID_KEY_F17 */	HID_KEY_F17 ,
122	/* HID_KEY_F18 */	HID_KEY_F18 ,
123	/* HID_KEY_F19 */	HID_KEY_F19 ,
124	/* HID_KEY_F20 */	HID_KEY_F20 ,
125	/* HID_KEY_F21 */	HID_KEY_F21 ,
126	/* HID_KEY_F22 */	HID_KEY_F22 ,
127	/* HID_KEY_F23 */	HID_KEY_F23 ,
128	/* HID_KEY_F24 */	HID_KEY_F24 ,
129	/* HID_KEY_EXECUTE */	HID_KEY_EXECUTE ,
130	/* HID_KEY_HELP */	HID_KEY_HELP ,
131	/* HID_KEY_MENU */	HID_KEY_MENU ,
132	/* HID_KEY_SELECT */	HID_KEY_SELECT ,
133	/* HID_KEY_STOP */	HID_KEY_STOP ,
134	/* HID_KEY_AGAIN */	HID_KEY_AGAIN ,
135	/* HID_KEY_UNDO */	HID_KEY_UNDO ,
136	/* HID_KEY_CUT */	HID_KEY_CUT ,
137	/* HID_KEY_COPY */	HID_KEY_COPY ,
138	/* HID_KEY_PASTE */	HID_KEY_PASTE ,
139	/* HID_KEY_FIND */	HID_KEY_FIND ,
140	/* HID_KEY_MUTE */	HID_KEY_MUTE ,
141	/* HID_KEY_VOLUME_UP */	HID_KEY_VOLUME_UP ,
142	/* HID_KEY_VOLUME_DOWN */	HID_KEY_VOLUME_DOWN ,
143	/* HID_KEY_LOCKING_CAPS_LOCK */	HID_KEY_LOCKING_CAPS_LOCK ,

144	/* HID_KEY_LOCKING_NUM_LOCK */	HID_KEY_LOCKING_NUM_LOCK ,
145	/* HID_KEY_LOCKING_SCROLL_LOCK */	HID_KEY_LOCKING_SCROLL_LOCK ,
146	/* HID_KEY_KEYPAD_COMMA */	HID_KEY_KEYPAD_COMMA ,
147	/* HID_KEY_KEYPAD_EQUAL_SIGN */	HID_KEY_KEYPAD_EQUAL_SIGN ,
148	/* HID_KEY_KANJI1 */	HID_KEY_KANJI1 ,
149	/* HID_KEY_KANJI2 */	HID_KEY_KANJI2 ,
150	/* HID_KEY_KANJI3 */	HID_KEY_KANJI3 ,
151	/* HID_KEY_KANJI4 */	HID_KEY_KANJI4 ,
152	/* HID_KEY_KANJI5 */	HID_KEY_KANJI5 ,
153	/* HID_KEY_KANJI6 */	HID_KEY_KANJI6 ,
154	/* HID_KEY_KANJI7 */	HID_KEY_KANJI7 ,
155	/* HID_KEY_KANJI8 */	HID_KEY_KANJI8 ,
156	/* HID_KEY_KANJI9 */	HID_KEY_KANJI9 ,
157	/* HID_KEY_LANG1 */	HID_KEY_LANG1 ,
158	/* HID_KEY_LANG2 */	HID_KEY_LANG2 ,
159	/* HID_KEY_LANG3 */	HID_KEY_LANG3 ,
160	/* HID_KEY_LANG4 */	HID_KEY_LANG4 ,
161	/* HID_KEY_LANG5 */	HID_KEY_LANG5 ,
162	/* HID_KEY_LANG6 */	HID_KEY_LANG6 ,
163	/* HID_KEY_LANG7 */	HID_KEY_LANG7 ,
164	/* HID_KEY_LANG8 */	HID_KEY_LANG8 ,
165	/* HID_KEY_LANG9 */	HID_KEY_LANG9 ,
166	/* HID_KEY_ALTERNATE_ERASE */	HID_KEY_ALTERNATE_ERASE ,
167	/* HID_KEY_SYSREQ_ATTENTION */	HID_KEY_SYSREQ_ATTENTION ,
168	/* HID_KEY_CANCEL */	HID_KEY_CANCEL ,
169	/* HID_KEY_CLEAR */	HID_KEY_CLEAR ,
170	/* HID_KEY_PRIOR */	HID_KEY_PRIOR ,
171	/* HID_KEY_RETURN */	HID_KEY_RETURN ,
172	/* HID_KEY_SEPARATOR */	HID_KEY_SEPARATOR ,
173	/* HID_KEY_OUT */	HID_KEY_OUT ,
174	/* HID_KEY_OPER */	HID_KEY_OPER ,
175	/* HID_KEY_CLEAR_AGAIN */	HID_KEY_CLEAR_AGAIN ,
176	/* HID_KEY_CRSEL_PROPS */	HID_KEY_CRSEL_PROPS ,
177	/* HID_KEY_EXSEL */	HID_KEY_EXSEL ,
178		RESERVED() ,
179		RESERVED() ,
180		RESERVED() ,
181		RESERVED() ,
182		RESERVED() ,
183		RESERVED() ,
184		RESERVED() ,
185		RESERVED() ,
186		RESERVED() ,
187		RESERVED() ,
188		RESERVED() ,
189	/* HID_KEY_KEYPAD_00 */	HID_KEY_KEYPAD_00 ,
190	/* HID_KEY_KEYPAD_000 */	HID_KEY_KEYPAD_000 ,
191	/* HID_KEY_THOUSANDS_SEPARATOR */	HID_KEY_THOUSANDS_SEPARATOR ,
192	/* HID_KEY_DECIMAL_SEPARATOR */	HID_KEY_DECIMAL_SEPARATOR ,
193	/* HID_KEY_CURRENCY_UNIT */	HID_KEY_CURRENCY_UNIT ,
194	/* HID_KEY_CURRENCY_SUBUNIT */	HID_KEY_CURRENCY_SUBUNIT ,
195	/* HID_KEY_KEYPAD_LEFT_PARENTHESIS */	HID_KEY_KEYPAD_LEFT_PARENTHESIS ,
196	/* HID_KEY_KEYPAD_RIGHT_PARENTHESIS */	HID_KEY_KEYPAD_RIGHT_PARENTHESIS ,
197	/* HID_KEY_KEYPAD_LEFT_BRACE */	HID_KEY_KEYPAD_LEFT_BRACE ,
198	/* HID_KEY_KEYPAD_RIGHT_BRACE */	HID_KEY_KEYPAD_RIGHT_BRACE ,
199	/* HID_KEY_KEYPAD_TAB */	HID_KEY_KEYPAD_TAB ,
200	/* HID_KEY_KEYPAD_BACKSPACE */	HID_KEY_KEYPAD_BACKSPACE ,
201	/* HID_KEY_KEYPAD_A */	HID_KEY_KEYPAD_A ,
202	/* HID_KEY_KEYPAD_B */	HID_KEY_KEYPAD_B ,
203	/* HID_KEY_KEYPAD_C */	HID_KEY_KEYPAD_C ,
204	/* HID_KEY_KEYPAD_D */	HID_KEY_KEYPAD_D ,
205	/* HID_KEY_KEYPAD_E */	HID_KEY_KEYPAD_E ,
206	/* HID_KEY_KEYPAD_F */	HID_KEY_KEYPAD_F ,
207	/* HID_KEY_KEYPAD_XOR */	HID_KEY_KEYPAD_XOR ,
208	/* HID_KEY_KEYPAD_CARET */	HID_KEY_KEYPAD_CARET ,
209	/* HID_KEY_KEYPAD_PERCENT */	HID_KEY_KEYPAD_PERCENT ,

```

210 /* HID_KEY_KEYPAD_LESS_THAN */      HID_KEY_KEYPAD_LESS_THAN ,
211 /* HID_KEY_KEYPAD_GREATER_THAN */    HID_KEY_KEYPAD_GREATER_THAN ,
212 /* HID_KEY_KEYPAD_AMPERSAND */       HID_KEY_KEYPAD_AMPERSAND ,
213 /* HID_KEY_KEYPAD_DOUBLE_AMPERSAND */ HID_KEY_KEYPAD_DOUBLE_AMPERSAND ,
214 /* HID_KEY_KEYPAD_VERTICAL_BAR */    HID_KEY_KEYPAD_VERTICAL_BAR ,
215 /* HID_KEY_KEYPAD_DOUBLE_VERTICAL_BAR */ HID_KEY_KEYPAD_DOUBLE_VERTICAL_BAR ,
216 /* HID_KEY_KEYPAD_COLON */           HID_KEY_KEYPAD_COLON ,
217 /* HID_KEY_KEYPAD_HASH */            HID_KEY_KEYPAD_HASH ,
218 /* HID_KEY_KEYPAD_SPACE */           HID_KEY_KEYPAD_SPACE ,
219 /* HID_KEY_KEYPAD_AT */              HID_KEY_KEYPAD_AT ,
220 /* HID_KEY_KEYPAD_EXCLAMATION */     HID_KEY_KEYPAD_EXCLAMATION ,
221 /* HID_KEY_KEYPAD_MEMORY_STORE */    HID_KEY_KEYPAD_MEMORY_STORE ,
222 /* HID_KEY_KEYPAD_MEMORY_RECALL */   HID_KEY_KEYPAD_MEMORY_RECALL ,
223 /* HID_KEY_KEYPAD_MEMORY_CLEAR */    HID_KEY_KEYPAD_MEMORY_CLEAR ,
224 /* HID_KEY_KEYPAD_MEMORY_ADD */      HID_KEY_KEYPAD_MEMORY_ADD ,
225 /* HID_KEY_KEYPAD_MEMORY_SUBTRACT */ HID_KEY_KEYPAD_MEMORY_SUBTRACT ,
226 /* HID_KEY_KEYPAD_MEMORY_MULTIPLY */ HID_KEY_KEYPAD_MEMORY_MULTIPLY ,
227 /* HID_KEY_KEYPAD_MEMORY_DIVIDE */   HID_KEY_KEYPAD_MEMORY_DIVIDE ,
228 /* HID_KEY_KEYPAD_PLUS_MINUS */      HID_KEY_KEYPAD_PLUS_MINUS ,
229 /* HID_KEY_KEYPAD_CLEAR */           HID_KEY_KEYPAD_CLEAR ,
230 /* HID_KEY_KEYPAD_CLEAR_ENTRY */     HID_KEY_KEYPAD_CLEAR_ENTRY ,
231 /* HID_KEY_KEYPAD_BINARY */          HID_KEY_KEYPAD_BINARY ,
232 /* HID_KEY_KEYPAD_OCTAL */           HID_KEY_KEYPAD_OCTAL ,
233 /* HID_KEY_KEYPAD_DECIMAL_2 */       HID_KEY_KEYPAD_DECIMAL_2 ,
234 /* HID_KEY_KEYPAD_HEXADEDECIMAL */   HID_KEY_KEYPAD_HEXADEDECIMAL ,
235 RESERVED() ,
236 RESERVED() ,
237 /* HID_KEY_CONTROL_LEFT */           HID_KEY_CONTROL_LEFT ,
238 /* HID_KEY_SHIFT_LEFT */             HID_KEY_SHIFT_LEFT ,
239 /* HID_KEY_ALT_LEFT */               HID_KEY_ALT_LEFT ,
240 /* HID_KEY_GUI_LEFT */              HID_KEY_GUI_LEFT ,
241 /* HID_KEY_CONTROL_RIGHT */          HID_KEY_CONTROL_RIGHT ,
242 /* HID_KEY_SHIFT_RIGHT */            HID_KEY_SHIFT_RIGHT ,
243 /* HID_KEY_ALT_RIGHT */              HID_KEY_ALT_RIGHT ,
244 /* HID_KEY_GUI_RIGHT */              HID_KEY_GUI_RIGHT ,
245 RESERVED() ,
246 RESERVED() ,
247 RESERVED() ,
248 RESERVED() ,
249 RESERVED() ,
250 RESERVED() ,
251 RESERVED() ,
252 RESERVED() ,
253 RESERVED() ,
254 RESERVED() ,
255 RESERVED() ,
256 RESERVED() ,
257 RESERVED() ,
258 RESERVED() ,
259 RESERVED() ,
260 RESERVED() ,
261 RESERVED() ,
262 RESERVED() ,
263 RESERVED() ,
264 RESERVED() ,
265 RESERVED() ,
266 RESERVED() ,
267 RESERVED() ,
268 RESERVED() ,
269 };
270
271
272 #endif /* _KBD_REMAP_H_ */

```

Listing 38: Remapper kbd_remap.h.

```

1  /*
2  * The MIT License (MIT)
3  *
4  * Copyright (c) 2019 Ha Thach (tinyusb.org)
5  *
6  * Permission is hereby granted, free of charge, to any person obtaining a copy
7  * of this software and associated documentation files (the "Software"), to deal
8  * in the Software without restriction, including without limitation the rights
9  * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
10 * copies of the Software, and to permit persons to whom the Software is
11 * furnished to do so, subject to the following conditions:
12 *
13 * The above copyright notice and this permission notice shall be included in
14 * all copies or substantial portions of the Software.
15 *
16 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
17 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
18 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
19 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
20 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
21 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
22 * THE SOFTWARE.
23 *
24 */
25
26 #include "bsp/board_api.h"
27 #include "tusb.h"
28 #include "usb_descriptors.h"
29
30 /* A combination of interfaces must have a unique product id, since PC will save
31    device driver after the first plug.
32    * Same VID/PID with different interface e.g MSC (first), then CDC (later) will
33    possibly cause system error on PC.
34    *
35    * Auto ProductID layout's Bitmap:
36    *   [MSB]          HID | MSC | CDC          [LSB]
37    */
38 #define _PID_MAP(itf, n)  ( (CFG_TUD_##itf) << (n) )
39 #define USB_PID          (0x4000 | _PID_MAP(CDC, 0) | _PID_MAP(MSC, 1) | _PID_MAP(
40    HID, 2) | \
41    _PID_MAP(MIDI, 3) | _PID_MAP(VENDOR, 4) )
42
43 #define USB_VID          0x2a2a
44 #define USB_BCD          0x0200
45
46 //-----+
47 // Device Descriptors
48 //-----+
49 tusb_desc_device_t const desc_device =
50 {
51     .bLength           = sizeof(tusb_desc_device_t),
52     .bDescriptorType    = TUSB_DESC_DEVICE,
53     .bcdUSB             = USB_BCD,
54     .bDeviceClass       = 0x00,
55     .bDeviceSubClass    = 0x00,
56     .bDeviceProtocol    = 0x00,
57     .bMaxPacketSize0    = CFG_TUD_ENDPOINT0_SIZE,
58
59     .idVendor           = USB_VID,
60     .idProduct          = USB_PID,
61     .bcdDevice          = 0x0100,
62
63     .iManufacturer      = 0x01,
64     .iProduct           = 0x02,
65     .iSerialNumber      = 0x03,
66

```

```

64     .bNumConfigurations = 0x01
65 };
66
67 // Invoked when received GET DEVICE DESCRIPTOR
68 // Application return pointer to descriptor
69 uint8_t const * tud_descriptor_device_cb(void)
70 {
71     return (uint8_t const *) &desc_device;
72 }
73
74 //-----+
75 // HID Report Descriptor
76 //-----+
77
78 uint8_t const desc_hid_report[] =
79 {
80     TUD_HID_REPORT_DESC_KEYBOARD()
81 #if 0
82     TUD_HID_REPORT_DESC_KEYBOARD( HID_REPORT_ID(REPORT_ID_KEYBOARD      )),
83     TUD_HID_REPORT_DESC_MOUSE    ( HID_REPORT_ID(REPORT_ID_MOUSE       )),
84     TUD_HID_REPORT_DESC_STYLUS_PEN( HID_REPORT_ID(REPORT_ID_STYLUS_PEN   )),
85     TUD_HID_REPORT_DESC_CONSUMER( HID_REPORT_ID(REPORT_ID_CONSUMER_CONTROL)),
86     TUD_HID_REPORT_DESC_GAMEPAD  ( HID_REPORT_ID(REPORT_ID_GAMEPAD      ))
87 #endif
88 };
89
90 // Invoked when received GET HID REPORT DESCRIPTOR
91 // Application return pointer to descriptor
92 // Descriptor contents must exist long enough for transfer to complete
93 uint8_t const * tud_hid_descriptor_report_cb(uint8_t instance)
94 {
95     (void) instance;
96     return desc_hid_report;
97 }
98
99 //-----+
100 // Configuration Descriptor
101 //-----+
102
103 enum
104 {
105     ITF_NUM_HID,
106     ITF_NUM_TOTAL
107 };
108
109 #define CONFIG_TOTAL_LEN  (TUD_CONFIG_DESC_LEN + TUD_HID_DESC_LEN)
110
111 #define EPNUM_HID  0x81
112
113 uint8_t const desc_configuration[] =
114 {
115     // Config number, interface count, string index, total length, attribute, power in
116     // mA
117     TUD_CONFIG_DESCRIPTOR(1, ITF_NUM_TOTAL, 0, CONFIG_TOTAL_LEN,
118         TUSB_DESC_CONFIG_ATT_REMOTE_WAKEUP, 100),
119     // Interface number, string index, protocol, report descriptor len, EP In address,
120     // size & polling interval
121     TUD_HID_DESCRIPTOR(ITF_NUM_HID, 0, HID_ITF_PROTOCOL_KEYBOARD, sizeof(
122         desc_hid_report), EPNUM_HID, CFG_TUD_HID_EP_BUFSIZE, 10)
123 };
124
125 #if TUD_OPT_HIGH_SPEED
126 #error highspeed
127 #endif // highspeed

```

```

126 // Invoked when received GET CONFIGURATION DESCRIPTOR
127 // Application return pointer to descriptor
128 // Descriptor contents must exist long enough for transfer to complete
129 uint8_t const * tud_descriptor_configuration_cb(uint8_t index)
130 {
131     (void) index; // for multiple configurations
132
133     // This example use the same configuration for both high and full speed mode
134     return desc_configuration;
135 }
136
137 //-----+
138 // String Descriptors
139 //-----+
140
141 // String Descriptor Index
142 enum {
143     STRID_LANGID = 0,
144     STRID_MANUFACTURER,
145     STRID_PRODUCT,
146     STRID_SERIAL,
147 };
148
149 // array of pointer to string descriptors
150 char const *string_desc_arr[] =
151 {
152     (const char[]) { 0x09, 0x04 }, // 0: is supported language is English (0x0409)
153     "CKAKL", // 1: Manufacturer
154     "CKAKL_Device", // 2: Product
155     NULL, // 3: Serials will use unique ID if possible
156 };
157
158 static uint16_t _desc_str[32 + 1];
159
160 // Invoked when received GET STRING DESCRIPTOR request
161 // Application return pointer to descriptor, whose contents must exist long enough
162 // for transfer to complete
163 uint16_t const *tud_descriptor_string_cb(uint8_t index, uint16_t langid) {
164     (void) langid;
165     size_t chr_count;
166
167     switch ( index ) {
168         case STRID_LANGID:
169             memcpy(&_desc_str[1], string_desc_arr[0], 2);
170             chr_count = 1;
171             break;
172
173         case STRID_SERIAL:
174             chr_count = board_usb_get_serial(_desc_str + 1, 32);
175             break;
176
177         default:
178             // Note: the 0xEE index string is a Microsoft OS 1.0 Descriptors.
179             // https://docs.microsoft.com/en-us/windows-hardware/drivers/usbcon/microsoft-
180             // defined-usb-descriptors
181
182             if ( !(index < sizeof(string_desc_arr) / sizeof(string_desc_arr[0])) ) return
183                 NULL;
184
185             const char *str = string_desc_arr[index];
186
187             // Cap at max char
188             chr_count = strlen(str);
189             size_t const max_count = sizeof(_desc_str) / sizeof(_desc_str[0]) - 1; // -1
190             // for string type
191             if ( chr_count > max_count ) chr_count = max_count;

```

```

188
189     // Convert ASCII string into UTF-16
190     for ( size_t i = 0; i < chr_count; i++ ) {
191         _desc_str[1 + i] = str[i];
192     }
193     break;
194 }
195
196 // first byte is length (including header), second byte is string type
197 _desc_str[0] = (uint16_t) ((TUSB_DESC_STRING << 8) | (2 * chr_count + 2));
198
199 return _desc_str;
200 }

```

Listing 39: Remapper usb_descriptors.c.

```

1  /*
2   * The MIT License (MIT)
3   *
4   * Copyright (c) 2019 Ha Thach (tinyusb.org)
5   *
6   * Permission is hereby granted, free of charge, to any person obtaining a copy
7   * of this software and associated documentation files (the "Software"), to deal
8   * in the Software without restriction, including without limitation the rights
9   * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
10  * copies of the Software, and to permit persons to whom the Software is
11  * furnished to do so, subject to the following conditions:
12  *
13  * The above copyright notice and this permission notice shall be included in
14  * all copies or substantial portions of the Software.
15  *
16  * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
17  * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
18  * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
19  * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
20  * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
21  * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
22  * THE SOFTWARE.
23  */
24
25 #ifndef USB_DESCRIPTOR_H_
26 #define USB_DESCRIPTOR_H_
27
28 enum
29 {
30     REPORT_ID_KEYBOARD = 1,
31 #if 0
32     REPORT_ID_MOUSE,
33     REPORT_ID_STYLUS_PEN,
34     REPORT_ID_CONSUMER_CONTROL,
35     REPORT_ID_GAMEPAD,
36 #endif
37     REPORT_ID_COUNT
38 };
39
40 #endif /* USB_DESCRIPTOR_H_ */

```

Listing 40: Remapper usb_descriptors.h.

```

1  /*
2   * The MIT License (MIT)
3   *
4   * Copyright (c) 2019 Ha Thach (tinyusb.org)
5   *
6   * Permission is hereby granted, free of charge, to any person obtaining a copy
7   * of this software and associated documentation files (the "Software"), to deal

```

```

8  * in the Software without restriction, including without limitation the rights
9  * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
10 * copies of the Software, and to permit persons to whom the Software is
11 * furnished to do so, subject to the following conditions:
12 *
13 * The above copyright notice and this permission notice shall be included in
14 * all copies or substantial portions of the Software.
15 *
16 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
17 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
18 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
19 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
20 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
21 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
22 * THE SOFTWARE.
23 *
24 */
25
26 #ifndef _TUSB_CONFIG_H_
27 #define _TUSB_CONFIG_H_
28
29 #ifdef __cplusplus
30     extern "C" {
31 #endif
32
33 //-----
34 // Board Specific Configuration
35 //-----
36
37 // RHPort number used for host can be defined by board.mk, default to port 0
38 #ifndef BOARD_TUH_RHPORT
39 #define BOARD_TUH_RHPORT        0
40 #endif
41
42 // RHPort number used for device can be defined by board.mk, default to port 1
43 #ifndef BOARD_TUD_RHPORT
44 #define BOARD_TUD_RHPORT        1
45 #endif
46
47 // RHPort max operational speed can defined by board.mk
48 #ifndef BOARD_TUH_MAX_SPEED
49 #define BOARD_TUH_MAX_SPEED     OPT_MODE_DEFAULT_SPEED
50 #endif
51
52 // RHPort max operational speed can defined by board.mk
53 #ifndef BOARD_TUD_MAX_SPEED
54 #define BOARD_TUD_MAX_SPEED     OPT_MODE_DEFAULT_SPEED
55 #endif
56
57 //-----
58 // Common Configuration
59 //-----
60
61 // defined by compiler flags for flexibility
62 #ifndef CFG_TUSB_MCU
63 #error CFG_TUSB_MCU must be defined
64 #endif
65
66 #ifndef CFG_TUSB_OS
67 #define CFG_TUSB_OS             OPT_OS_NONE
68 #endif
69
70 #ifndef CFG_TUSB_DEBUG
71 #define CFG_TUSB_DEBUG          0
72 #endif
73

```

```

74 // Enable Host stack, Default is max speed that hardware controller could support
    with on-chip PHY
75 #define CFG_TUH_ENABLED          1
76 #define CFG_TUH_MAX_SPEED        BOARD_TUH_MAX_SPEED
77
78 // Enable Device stack, Default is max speed that hardware controller could support
    with on-chip PHY
79 #define CFG_TUD_ENABLED          1
80 #define CFG_TUD_MAX_SPEED        BOARD_TUD_MAX_SPEED
81
82 /* USB DMA on some MCUs can only access a specific SRAM region with restriction on
    alignment.
83 * Tinyusb use follows macros to declare transferring memory so that they can be put
84 * into those specific section.
85 * e.g
86 * - CFG_TUSB_MEM_SECTION : __attribute__((section(".usb_ram")))
87 * - CFG_TUSB_MEM_ALIGN   : __attribute__((aligned(4)))
88 */
89 #ifndef CFG_TUD_MEM_SECTION
90 #define CFG_TUD_MEM_SECTION
91 #endif
92
93 #ifndef CFG_TUD_MEM_ALIGN
94 #define CFG_TUD_MEM_ALIGN        __attribute__((aligned(4)))
95 #endif
96
97 #ifndef CFG_TUH_MEM_SECTION
98 #define CFG_TUH_MEM_SECTION
99 #endif
100
101 #ifndef CFG_TUH_MEM_ALIGN
102 #define CFG_TUH_MEM_ALIGN        __attribute__((aligned(4)))
103 #endif
104
105 //-----
106 // Device Configuration
107 //-----
108
109 #ifndef CFG_TUD_ENDPOINT0_SIZE
110 #define CFG_TUD_ENDPOINT0_SIZE   64
111 #endif
112
113 //----- CLASS -----//
114 #define CFG_TUD_HID              1
115 #define CFG_TUD_CDC              0
116 #define CFG_TUD_MSC              0
117 #define CFG_TUD_MIDI            0
118 #define CFG_TUD_VENDOR          0
119
120 #define CFG_TUD_HID_EP_BUFSIZE   16
121
122 //-----
123 // Host Configuration
124 //-----
125
126 // Size of buffer to hold descriptors and other data used for enumeration
127 #define CFG_TUH_ENUMERATION_BUFSIZE 256
128
129 #define CFG_TUH_HUB              0
130 #define CFG_TUH_CDC              0
131 #define CFG_TUH_HID              (3*CFG_TUH_DEVICE_MAX) // typical keyboard +
    mouse device can have 3-4 HID interfaces
132 #define CFG_TUH_MSC              0
133 #define CFG_TUH_VENDOR          0
134
135 // max device support (excluding hub device): 1 hub typically has 4 ports

```

```

136 #define CFG_TUH_DEVICE_MAX          (3*CFG_TUH_HUB + 1)
137
138 #ifdef __cplusplus
139 }
140 #endif
141
142 #endif /* _TUSB_CONFIG_H_ */

```

Listing 41: Remapper `tusb_config.h`.

B.4 YAML parser

```

1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4
5  #include <cyaml/cyaml.h>
6
7  /* Structure for storing the keyboard mappings.
8   * @count: The amount of remappings stored in the structure.
9   * @from: The key that is being remapped.
10  * @to: The key press that should be sent instead.
11  */
12
13 struct mapping {
14     unsigned count;
15     char** from;
16     char** to;
17 };
18
19 /* Structure for storing the data from the YAML file (from libcyaml parsing).
20  * @did: The "device_id" field.
21  * @vid: The "vendor_id" field.
22  * @raw_mappings_count: How many raw mappings have been read from the file.
23  * @raw_mappings: An array of strings of all the mappings present in the file.
24  */
25 struct yaml_file {
26     unsigned int did;
27     unsigned int vid;
28     unsigned int raw_mappings_count;
29     char** raw_mappings;
30 };
31
32 // "libcyaml" macro for defining the mappings as a string of an "unlimited" length.
33 static const cyaml_schema_value_t mapping_entry = {
34     CYAML_VALUE_STRING(CYAML_FLAG_POINTER, char, 0, CYAML_UNLIMITED),
35 };
36
37 /* "libcyaml" macros for defining the top level schema (layout) of the YAML file.
38  * Defining "device_id" as a unsigned integer value and storing it in the "yaml_file"
39  * structure.
40  * Defining "vendor_id" as a unsigned integer value and storing it in the "yaml_file"
41  * structure.
42  * Defining "mappings" as a sequence of strings and the length being optional and
43  * storing the data in the "yaml_file" structure.
44  */
45 static const cyaml_schema_field_t top_mapping_schema[] = {
46     CYAML_FIELD_UINT("device_id", CYAML_FLAG_DEFAULT, struct yaml_file, did),
47     CYAML_FIELD_UINT("vendor_id", CYAML_FLAG_DEFAULT, struct yaml_file, vid),
48     CYAML_FIELD_SEQUENCE("mappings", CYAML_FLAG_POINTER | CYAML_FLAG_OPTIONAL, struct
49         yaml_file, raw_mappings,
50         &mapping_entry, 0, CYAML_UNLIMITED),
51     CYAML_FIELD_END
52 };

```

```

50 // "libcyaml" macro mapping the top level schema to the file.
51 static const cyaml_schema_value_t top_schema = {
52     CYAML_VALUE_MAPPING(CYAML_FLAG_POINTER, struct yaml_file, top_mapping_schema),
53 };
54
55 // Configuration options for the "libcyaml" library.
56 static const cyaml_config_t config = {
57     .log_fn = cyaml_log,          /* Use the default logging function. */
58     .mem_fn = cyaml_mem,          /* Use the default memory allocator. */
59     .log_level = CYAML_LOG_WARNING, /* Logging errors and warnings only. */
60 };
61
62 /* Function to iterate over the raw strings read from the YAML configuration file
63  * and parse them into the "mappings" structure. It splits the strings on the ':'
64  * delimiter and stores the two tokens into the structure, allocating memory where
65  * appropriate.
66  */
67 void parse_raw_mappings(struct yaml_file* file, struct mapping* mapping) {
68     char* tmp;
69
70     // allocate memory for the pointers to strings
71     mapping->from = malloc(file->raw_mappings_count * sizeof(char*));
72     mapping->to = malloc(file->raw_mappings_count * sizeof(char*));
73     for (unsigned i = 0; i < file->raw_mappings_count; i++) {
74         // split string on the ':' delimiter
75         tmp = strtok(file->raw_mappings[i], ":");
76         // allocate memory for the first token and copy the string
77         mapping->from[i] = malloc(strlen(tmp) + 1);
78         strcpy(mapping->from[i], tmp);
79         // move onto the next token (right hand side of ':')
80         tmp = strtok(NULL, ":");
81         // allocate memory for the token and copy the string
82         mapping->to[i] = malloc(strlen(tmp) + 1);
83         strcpy(mapping->to[i], tmp);
84     }
85     mapping->count = file->raw_mappings_count;
86 }
87
88 /* Debug function to print out the data that has been read and parsed from the YAML
89  * configuration file.
90  */
91 void print_mappings(const struct mapping* mapping, const struct yaml_file* yf) {
92     printf("Device_ID: %u\n", yf->did);
93     printf("Vendor_ID: %u\n", yf->vid);
94     printf("Mappings:\n");
95     for (unsigned i = 0; i < mapping->count; i++) {
96         printf("\tMap_Key(%s) to (%s)\n", mapping->from[i], mapping->to[i]);
97     }
98 }
99
100 int main(void) {
101     cyaml_err_t err;
102     struct yaml_file *yf;
103     struct mapping mapping;
104
105     // load YAML file from "config.yaml"
106     // TODO: remove hardcoded path
107     err = cyaml_load_file("config.yaml", &config, &top_schema, (cyaml_data_t **)&yf,
108         NULL);
109     // check for errors
110     if (err != CYAML_OK) {
111         // print error and exit program
112         fprintf(stderr, "ERROR: %s\n", cyaml_strerror(err));
113         return EXIT_FAILURE;
114     }
115     // retrieve mappings from strings and store in "mapping"

```

```
115 | parse_raw_mappings(yf, &mapping);
116 |
117 | // DEBUG: print the data read from YAML file and "mapping"
118 | print_mappings(&mapping, yf);
119 |
120 | return 0;
121 | }
```

Listing 42: YAML main.c.

C Log

The following log was adapted from the written version kept by *JG*.

C.1 Week 1

C.1.1 19/05/25

- Decide on USB modifier *BG,JG,SH*
- Come up with problem statement *BG,JG,SH*
- Research USB stack *JG*
- Research file formats for configuration *BG*
- Research regulations/legality *BG*
- Research ICs microcontroller etc. *SH*
- Decide on STM32F407VGT6 as micro-controller - has 2 USB ports *SH,BG*
- Micro-controller uses 110 mA of 500 mA provided by USB 2.0 *BG*
- Research keyboard+scancodes *BG*
- Initial Gantt chart *BG,JG,SH*
- Research flash IC *SH*
- Research USB passthrough with file system in parallel *JG*

C.1.2 20/05/25

- Research written up *BG,SH*
- Perform literature review *JG*
- Produce draft specification *BG,SH*
- Editorial work *JG*

C.1.3 21/05/25

- Start methodology writeup *BG*
- Start PCB design *SH*
- Decide on USB layout *JG,SH*
- Choose 8 MHz clock *JG,SH*
- Add button debouncing filter *JG,SH*
- Prepare presentation for meeting *BG,JG,SH*
- Editorial work *JG*

C.1.4 22/05/25

- Schematic mostly done *SH*
- Meeting with Jenifer and Ben – happy for us to proceed *BG,JG,SH*
- Start PCB design *SH*
- Gather component footprints *JG,SH*
- Research YAML parsing *BG*

C.2 Week 2

C.2.1 27/05/25

- Discussion with senior engineer discussing schematic and PCB: feedback on component size, spacing, pins, and design choices *BG,JG,SH*

C.2.2 30/05/25

- Applied feedback to schematic and PCB done *SH*
- Final checkover by senior engineer *JG,SH*

C.3 Week 3

C.3.1 02/06/25

- Order PCB and additional components *BG,JG,SH*

C.3.2 06/06/25

- Update Gantt chart *SH*
- Research suitable USB stack *JG*

C.4 Week 4

C.4.1 12/06/25

- Received the PCB *BG,JG,SH*
- Soldered remaining components *SH*
- Attempted to flash SWD test program - didn't work *JG,SH*
- Add jumper wire to correct the SWD header *SH*
- Second flash of SWD test program worked *JG*
- Research YAML libraries *BG*

C.4.2 13/06/25

- Tested LED and switch programs *JG*

C.5 Week 5

C.5.1 15/06/25

- Add serial wire *SH*
- Test serial *JG*

C.5.2 16/06/25

- Test SPI interface *JG*
- Test flash read and write *JG,SH*

C.5.3 20/06/25

- USB (Device side) D+ and D- wrong way round *JG,SH*
- Add jumper wires to fix *SH*

C.6 Week 6

C.6.1 24/06/25

- Fix pull-up resistor for USB (Device side) *JG,SH*
- Decide to redesign PCB *BG,JG,SH*
- Add fixed to schematic *SH*
- Implement YAML logic *BG*

C.6.2 25/06/25

- Tested keyboard input *BG,JG,SH*

C.6.3 26/06/25

- Write passthrough for keyboard to computer *JG*

C.6.4 27/06/25

- Finished redesign of PCB *SH*
- Order second batch of PCBs *BG,JG,SH*
- Finish writing CKAKL program with configuration *JG*

C.7 Week 7

C.7.1 03/07/25

- Received second PCB *BG,JG,SH*
- Solder the new PCB *SH*
- Test new PCB *BG,JG,SH*
- CKAKL program works on new PCB *JG*
- Implement string tokenisation *BG*

C.8 Week 8

Report was written *BG,JG,SH*